**NIST Special Publication XXX-XXX**

# DRAFT NIST Big Data Interoperability Framework: Volume 7, Technology Roadmap

NIST Big Data Public Working Group
Technology Roadmap Subgroup

Draft Version 1
April 23, 2014

http://dx.doi.org/10.6028/NIST.SP.XXX

National Institute of Standards and Technology
U.S. Department of Commerce

NIST Special Publication xxx-xxx
Information Technology Laboratory

# DRAFT NIST Big Data Interoperability Framework: Volume 7, Technology Roadmap

**Version 1**

NIST Big Data Public Working Group (NBD-PWG)
Technology Roadmap Subgroup
National Institute of Standards and Technology
Gaithersburg, MD 20899

Month 2014

U. S. Department of Commerce
*Penny Pritzker, Secretary*

*National Institute of Standards and Technology*
*Patrick D. Gallagher, Under Secretary of Commerce for Standards and Technology and Director*

## Authority

This publication has been developed by National Institute of Standards and Technology (NIST) to further its statutory responsibilities …

Nothing in this publication should be taken to contradict the standards and guidelines made mandatory and binding on Federal agencies ….

**Comments on this publication may be submitted to:**
National Institute of Standards and Technology
Attn: Information Technology Laboratory
100 Bureau Drive (Mail Stop 8900) Gaithersburg, MD 20899-8930

## Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in Federal information systems. This document reports on ITL's research, guidance, and outreach efforts in Information Technology and its collaborative activities with industry, government, and academic organizations.

**National Institute of Standards and Technology Special Publication XXX-series**
xxx pages (April 23, 2014)

## Acknowledgements

This document reflects the contributions and discussions by the membership of the NIST Big Data Public Working Group (NBD-PWG), co-chaired by Wo Chang of the NIST Information Technology Laboratory, Robert Marcus of ET-Strategies, and Chaitanya Baru, University of California San Diego Supercomputer Center.

The document contains input from members of the NBD-PWG: Technology Roadmap Subgroup, led by Carl Buffington (Vistronix), David Boyd (Data Tactic), and Dan McClary (Oracle); Definitions and Taxonomies Subgroup, led by Nancy Grady (SAIC), Natasha Balac (SDSC), and Eugene Luster (R2AD); Use Cases and Requirements Subgroup, led by Geoffrey Fox (University of Indiana) and Tsegereda Beyene (Cisco); Security and Privacy Subgroup, led by Arnab Roy (Fujitsu) and Akhil Manchanda (GE); and Reference Architecture Subgroup, led by Orit Levin (Microsoft), Don Krapohl (Augmented Intelligence), and James Ketner (AT&T).

NIST SP xxx-series, Version 1 has been collaboratively authored by the NBD-PWG. As of the date of this publication, there are over six hundred NBD-PWG participants from industry, academia, and government. Federal agency participants include the National Archives and Records Administration (NARA), National Aeronautics and Space Administration (NASA), National Science Foundation (NSF), and the U.S. Departments of Agriculture, Commerce, Defense, Energy, Health and Human Services, Homeland Security, Transportation, Treasury, and Veterans Affairs.

NIST would like to acknowledge the specific contributions to this volume by the following NBD-PWG members:

Chaitan Baru, University of California, San Diego, Supercomputer Center

David Boyd, Data Tactics Corporation

Carl Buffington, Vistronix

Pw Carey, Compliance Partners, LLC

Wo Chang, National Institute of Standards and Technology

Yuri Demchenko, University of Amsterdam

Nancy Grady, SAIC

Keith Hare, JCC Consulting, Inc.

Bruno Kelpsas, Microsoft Consultant

Pavithra Kenjige, PK Technologies

Brenda Kirkpatrick, Hewlett-Packard

Donald Krapohl, Augmented Intelligence

Luca Lepori, Data Hold

Orit Levin, Microsoft

Jan Levine, kloudtrack

Serge Mankovski, CA Technologies

Robert Marcus, ET-Strategies

Gary Mazzaferro, AlloyCloud, Inc.

Shawn Miller, U.S. Department of Veterans Affairs

William Miller, MaCT USA

Sanjay Mishra, Verizon

Quyen Nguyen, National Archives and Records Administration

John Rogers, HP

Doug Scrimager, Slalom Consulting

Cherry Tom, IEEE-SA

Timothy Zimmerlin, Automation Technologies Inc.

The editors for this document were David Boyd, Carl Buffington, and Wo Chang.

# Table of Contents

## Figures

## Tables

# Executive Summary

This ***NIST Big Data Interoperability Framework: Volume 7, Technology Roadmap*** was prepared by the NBD-PWG's Technology Roadmap Subgroup. It draws on the other volumes in the set and adds overarching information and context about these key questions:

- When is data considered "Big"?
- How did Big Data evolve?
- What will it evolve to?
- How is technology developing to deal with Big Data in terms of storage, organization, processing, and resource management?
- What standards are needed and evolving to deal with Big Data? and,
- How might organizations address their Big Data challenges?

There is broad agreement among commercial, academic, and government leaders about the remarkable potential of Big Data to spark innovation, fuel commerce, and drive progress. The big question—often referred to as the big problem—about Big Data is: how do we do this? This volume seeks to address Big Data technology from the following perspectives:

- **Organization readiness** – Is the organization mature in the right ways to adopt and make use of Big Data (or to know if they even have a Big Data problem)
- **Technology Readiness** – how can one assess if a given technology is mature enough and ready to address a specific Big Data problem
- **Big Data Technology features** – What are the core feature classes of Big Data technologies, how are they evolving, and how does one assess their applicability to a specific problem.
- **Big Data Standards initiatives** – What standards for portability, interoperability, etc. are or need to evolve for Big Data.
- **Big Data Strategies** – What strategies are evolving to adopt or deal with Big Data and what are the roadblocks to adoption of Big Data solutions.

By design, this document contains various technical and managerial perspectives, all of which are essential to addressing Big Data challenges and opportunities. However, also by design, it is written so that each reader can focus on the topics of greatest interest to them. For example, if your perspective is mostly technical, the features perspectives may be of the most use to you. Or, if you are a manager or executive looking to adopt Big Data solutions, the organization readiness and strategies will likely be most applicable.

The following responsibility assignment matrix[1] was developed to give readers an idea of the roles and responsibilities most likely involved in implementing the material in this document.

**R**esponsible

**A**ccountable (also approver or final approving authority)

**C**onsulted (sometimes counsel)

**I**nformed

| | **Executive Stakeholders** | **Technical Architects and Managers** | **Quantitative Roles** | **Application Development** | **Systems Operation & Administration** |
|---|---|---|---|---|---|
| Organizational Adoption and Business Strategy | **R** | **A** | **C** | **C** | **I** |
| Infrastructure and Architecture | **I** | **R** | **C** | **A** | **A** |
| Complex analytics, reporting, and business intelligence | **C** | **A** | **R** | **A** | **I** |
| Programming paradigms and information management | **I** | **A** | **C** | **R** | **A** |
| Deployment, administration, and maintenance | **I** | **A** | **C** | **A** | **R** |

***Figure 1: Roadmap Content as Responsibility Assignment Matrix.***

Because the technology of Big Data and the processes for adopting/leveraging Big Data are evolving rapidly, this document is by definition incomplete and will likely always be behind what is available. It is the authors' hope that this document will grow and evolve as the Big Data business and technical ecosystems evolve, and that the community will continue to help us in that effort.

The following are the preliminary consensus working drafts of the NIST Big Data Interoperability Framework:

- Volume 1: Definitions
- Volume 2: Taxonomies
- Volume 3: Use Cases and General Requirements
- Volume 4: Security and Privacy Requirements
- Volume 5: Architectures White Paper Survey
- Volume 6: Reference Architectures
- Volume 7: Technology Roadmap

# 1 Introduction

“Moore's law”—which was actually an observation by Intel co-founder Gordon E. Moore, who described the trend in his 1965 paper—is that, over the history of computing hardware, the number of transistors on integrated circuits doubles approximately every two years. His prediction has continued to prove itself, and has been strongly linked to the capabilities of many digital electronic devices: processing speed, memory capacity, sensors, and even the number and size of pixels in digital cameras. This exponential improvement has dramatically enhanced the impact of digital electronics in nearly every segment of the world economy.

There is an old saying that everything is about perspective. The fundamental description of Big Data is that it is too big (volume), or arrives too fast(velocity), or is too diverse (variety) to be processed within a local computing structure without using additional approaches/techniques to make the data fit or provide a result in an acceptable time frame. If we look at Big Data from a time perspective, what was considered extremely large even five years ago can be handled easily today on portable and mobile platforms. The use of swapping and paging from ram to disk or other media was one of the very first techniques employed to deal with what was thought of as Big Data years ago. What will be considered big five years from now may likely depend on how well Moore’s Law continues to hold. From a connectivity perspective, what is considered big is determined by how long it would take to retrieve/move the data to get an answer or in some cases if it would be even possible to move the data. A high-resolution image from a sensor would likely not be considered big when being retrieved and processed within a data center or even office-networking environment.

However, to a soldier on dismounted patrol in the mountains of Afghanistan it is not even practical to being to transfer him that data in its native form. From a variety or complexity perspective, even our cell phones today easily process a wide variety of web-based content, including text and video. On the other side, there is too much variety in that data for the processor to reason about it or turn it into relevant information and knowledge without a human reviewing it. Even large data centers struggle to align, and reason about diverse data and the long-term vision of a semantic web[2] must deal heavily with the diverse domains and multiple semantic meanings assigned to common terms and concepts.

The total scale of data is well described by the NSA an organization that has dealt with Big Data type problems for decades as follows "According to figures published by a major tech provider, the Internet carries 1,826 Petabytes of information per day. In its foreign intelligence mission, NSA touches about 1.6% of that. However, of the 1.6% of the data, only 0.025% is actually selected for review. The net effect is that NSA analysts look at 0.00004% of the world's traffic in conducting their mission - that is less than one part in a million. Put another way, if a standard basketball court represented the global communications environment, NSA's total collection would be represented by an area smaller than a dime on that basketball court."

Essentially the problem with Big Data is that at some point there is a threshold at which for a certain set of data and specific application at which simply using a faster processor, more memory, more storage, or other traditional data management techniques (scaling vertically, data Organization/indexing, algorithms) cannot produce an answer in an acceptable timeframe and requires approaches that distribute the data across multiple processing nodes (scale horizontally) to meet the application requirements.

## 1.1 Background

There is broad agreement among commercial, academic, and government leaders about the remarkable potential of Big Data to spark innovation, fuel commerce, and drive progress. Big Data is the common term used to describe the deluge of data in our networked, digitized, sensor-laden, information-driven world. The availability of vast data resources carries the potential to answer questions previously out of reach, including the following:

- How can we reliably detect a potential pandemic early enough to intervene?
- Can we predict new materials with advanced properties before these materials have ever been synthesized?
- How can we reverse the current advantage of the attacker over the defender in guarding against cyber-security threats?

However, there is also broad agreement on the ability of Big Data to overwhelm traditional approaches. The growth rates for data volumes, speeds, and complexity are outpacing scientific and technological advances in data analytics, management, transport, and data user spheres.

Despite the widespread agreement on the inherent opportunities and current limitations of Big Data, a lack of consensus on some important, fundamental questions continues to confuse potential users and stymie progress. These questions include the following:

- What attributes define Big Data solutions?
- How is Big Data different from traditional data environments and related applications?
- What are the essential characteristics of Big Data environments?
- How do these environments integrate with currently deployed architectures?
- What are the central scientific, technological, and standardization challenges that need to be addressed to accelerate the deployment of robust Big Data solutions?

Within this context, on March 29, 2012, the White House announced the Big Data Research and Development Initiative.[3] The initiative's goals include helping to accelerate the pace of discovery in science and engineering, strengthening national security, and transforming teaching and learning by improving our ability to extract knowledge and insights from large and complex collections of digital data.

Six federal departments and their agencies announced more than $200 million in commitments spread across more than 80 projects, which aim to significantly improve the tools and techniques needed to access, organize, and draw conclusions from huge volumes of digital data. The initiative also challenged industry, research universities, and nonprofits to join with the federal government to make the most of the opportunities created by Big Data.

Motivated by the White House's initiative and public suggestions, the National Institute of Standards and Technology (NIST) has accepted the challenge to stimulate collaboration among industry professionals to further the secure and effective adoption of Big Data. As one result of NIST's Cloud and Big Data Forum held January 15–17, 2013, there was strong encouragement for NIST to create a public working group for the development of a Big Data Interoperability Framework. Forum participants noted that this roadmap should define and prioritize Big Data requirements, including interoperability, portability, reusability, extensibility, data usage, analytics, and technology infrastructure. In doing so, the roadmap would accelerate the adoption of the most secure and effective Big Data techniques and technology.

On June 19, 2013, the NIST Big Data Public Working Group (NBD-PWG) was launched with overwhelming participation from industry, academia, and government from across the nation. The scope of the NBD-PWG involves forming a community of interests from all sectors—including industry, academia, and government—with the goal of developing a consensus on definitions, taxonomies, secure reference architectures, security and privacy requirements, and a technology roadmap. Such a consensus would create a vendor-neutral, technology- and infrastructure-independent framework that would enable Big Data stakeholders to identify and use the best analytics tools for their processing and visualization requirements on the most suitable computing platform and cluster, while also allowing value-added from Big Data service providers.

## 1.2 Scope and Objectives of the Technology Roadmap Subgroup

The Technology Roadmap Subgroup focused on forming a community of interest from industry, academia, and government, with the goal of developing a consensus vision with recommendations on how Big Data should move forward. The subgroup's approach was to perform a gap analysis through the materials gathered from all other subgroups. This included setting standardization and adoption priorities through an understanding of what standards are available or under development as part of the recommendations. The primary tasks of the Technology Roadmap Subgroup included:

- Gather input from NBD subgroups and study the taxonomies for the actors' roles and responsibility, use cases and requirements, and secure reference architecture.
- Gain understanding of what standards are available or under development for Big Data.
- Perform a thorough gap analysis and document the findings.
- Identify what possible barriers may delay or prevent adoption of Big Data.
- Document vision and recommendations.

## 1.3 Production of this Report

This document and its companions were developed based on the following guiding principles.

- It is technologically agnostic.
- The audience is multi-sector: industry, government, and academia.
- It is aligned with the findings of the other subgroups.
- It is the culmination of concepts from all subgroups.
- It recommends actionable items for Big Data programs.

## 1.4 Structure of This Report

This volume contains five major components:

Chapter 1: This introduction.

Chapters 2-6: Summary material from the topic-specific volumes in the set, covering the key topics of: definitions; taxonomies; use cases and general requirements; security and privacy requirements; and reference architecture.

Chapter 7: Assessment of the technology and organizational readiness and features that are the main elements involved in how Big Data is addressed.

Chapter 8: Overview of ongoing, major multi-stakeholder collaborative initiatives that are related to Big Data that can be leveraged to advance Big Data solutions

Chapter 9: A beginning discussion of action plans to address Big Data challenges.

A brief description of future directions is also provided.

# 2 Big Data Definitions

<Content—a summary of Roadmap Volume 1—is under development.>

# 3 Big Data Taxonomies

<Content—a summary of Roadmap Volume 2—is under development.>

# 4 Big Data Use Cases and General Requirements

Requirements are the challenges limiting further use of Big Data. After collection, processing, and review of the use cases, requirements within seven characteristic categories were extracted from the individual use cases. These use case specific requirements were then aggregated to produce high-level, general requirements, within the seven characteristic categories, that are vendor neutral and technology agnostic. Neither the use case nor the requirements lists are exhaustive.

The data are presented online at the following links:

- Index to all use cases: http://bigdatawg.nist.gov/usecases.php
- List of specific requirements versus use case: http://bigdatawg.nist.gov/uc_reqs_summary.php
- List of general requirements versus architecture component: http://bigdatawg.nist.gov/uc_reqs_gen.php
- List of general requirements versus architecture component with record of use cases giving requirements: http://bigdatawg.nist.gov/uc_reqs_gen_ref.php
- List of architecture components and specific requirements plus use case constraining the components: http://bigdatawg.nist.gov/uc_reqs_gen_detail.php

General requirements can be obtained from http://bigdatawg.nist.gov/uc_reqs_gen.php.

Each use case was evaluated for requirements within the following seven categories:

- Data sources (e.g., data size, file formats, rate of growth, at rest or in motion)
- Data transformation (e.g., data fusion, analytics)
- Capabilities (e.g., software tools, platform tools, hardware resources such as storage and networking)
- Data consumer (e.g., processed results in text, table, visual, and other formats)
- Security and Privacy
- Lifecycle management (curation, conversion, quality check, pre-analytic processing, etc.)
- Other requirements

Some use cases contained requirements in all seven categories while others only produced requirements for a few categories. The complete list of requirements extracted from the use cases is presented in Appendix D.

## 4.1 General Requirements

<Introduction under development.>

**Government Operation**

1. Census 2010 and 2000 – Title 13 Big Data; Vivek Navale & Quyen Nguyen, NARA
2. National Archives and Records Administration Accession NARA, Search, Retrieve, Preservation; Vivek Navale & Quyen Nguyen, NARA

**Commercial**

3. Cloud Eco-System, for Financial Industries (Banking, Securities & Investments, Insurance) transacting business within the United States; Pw Carey, Compliance Partners, LLC
4. Mendeley – An International Network of Research; William Gunn , Mendeley
5. Netflix Movie Service; Geoffrey Fox, Indiana University
6. Web Search; Geoffrey Fox, Indiana University

7. IaaS (Infrastructure as a Service) Big Data Business Continuity & Disaster Recovery (BC/DR) Within A Cloud Eco-System; Pw Carey, Compliance Partners, LLC
8. Cargo Shipping; William Miller, MaCT USA
9. Materials Data for Manufacturing; John Rumble, R&R Data Services
10. Simulation driven Materials Genomics; David Skinner, LBNL

**Healthcare and Life Sciences**

11. Electronic Medical Record (EMR) Data; Shaun Grannis, Indiana University
12. Pathology Imaging/digital pathology; Fusheng Wang, Emory University
13. Computational Bioimaging; David Skinner, Joaquin Correa, Daniela Ushizima, Joerg Meyer, LBNL
14. Genomic Measurements; Justin Zook, NIST
15. Comparative analysis for metagenomes and genomes; Ernest Szeto, LBNL (Joint Genome Institute)
16. Individualized Diabetes Management; Ying Ding , Indiana University
17. Statistical Relational Artificial Intelligence for Health Care; Sriraam Natarajan, Indiana University
18. World Population Scale Epidemiological Study; Madhav Marathe, Stephen Eubank or Chris Barrett, Virginia Tech
19. Social Contagion Modeling for Planning, Public Health and Disaster Management; Madhav Marathe or Chris Kuhlman, Virginia Tech
20. Biodiversity and LifeWatch; Wouter Los, Yuri Demchenko, University of Amsterdam

**Deep Learning and Social Media**

21. Large-scale Deep Learning; Adam Coates , Stanford University
22. Organizing large-scale, unstructured collections of consumer photos; David Crandall, Indiana University
23. Truthy: Information diffusion research from Twitter Data; Filippo Menczer, Alessandro Flammini, Emilio Ferrara, Indiana University
24. CINET: Cyberinfrastructure for Network (Graph) Science and Analytics; Madhav Marathe or Keith Bisset, Virginia Tech
25. NIST Information Access Division analytic technology performance measurement, evaluations, and standards; John Garofolo, NIST

**The Ecosystem for Research**

26. DataNet Federation Consortium DFC; Reagan Moore, University of North Carolina at Chapel Hill
27. The 'Discinnet process', metadata <-> big data global experiment; P. Journeau, Discinnet Labs
28. Semantic Graph-search on Scientific Chemical and Text-based Data; Talapady Bhat, NIST
29. Light source beamlines; Eli Dart, LBNL

**Astronomy and Physics**

30. Catalina Real-Time Transient Survey (CRTS): a digital, panoramic, synoptic sky survey; S. G. Djorgovski, Caltech
31. DOE Extreme Data from Cosmological Sky Survey and Simulations; Salman Habib, Argonne National Laboratory; Andrew Connolly, University of Washington
32. Particle Physics: Analysis of LHC Large Hadron Collider Data: Discovery of Higgs particle; Geoffrey Fox, Indiana University; Eli Dart, LBNL

**Earth, Environmental and Polar Science**

33. EISCAT 3D incoherent scatter radar system; Yin Chen, Cardiff University; Ingemar Häggström, Ingrid Mann, Craig Heinselman, EISCAT Science Association
34. ENVRI, Common Operations of Environmental Research Infrastructure; Yin Chen, Cardiff University
35. Radar Data Analysis for CReSIS Remote Sensing of Ice Sheets; Geoffrey Fox, Indiana University
36. UAVSAR Data Processing, Data Product Delivery, and Data Services; Andrea Donnellan and Jay Parker, NASA JPL
37. NASA LARC/GSFC iRODS Federation Testbed; Brandi Quam, NASA Langley Research Center
38. MERRA Analytic Services MERRA/AS; John L. Schnase & Daniel Q. Duffy , NASA Goddard Space Flight Center
39. Atmospheric Turbulence - Event Discovery and Predictive Analytics; Michael Seablom, NASA HQ
40. Climate Studies using the Community Earth System Model at DOE's NERSC center; Warren Washington, NCAR
41. DOE-BER Subsurface Biogeochemistry Scientific Focus Area; Deb Agarwal, LBNL
42. DOE-BER AmeriFlux and FLUXNET Networks; Deb Agarwal, LBNL

# 5 Big Data Security and Privacy Requirements

<Content—a summary of Roadmap Volume 4—is under development.>

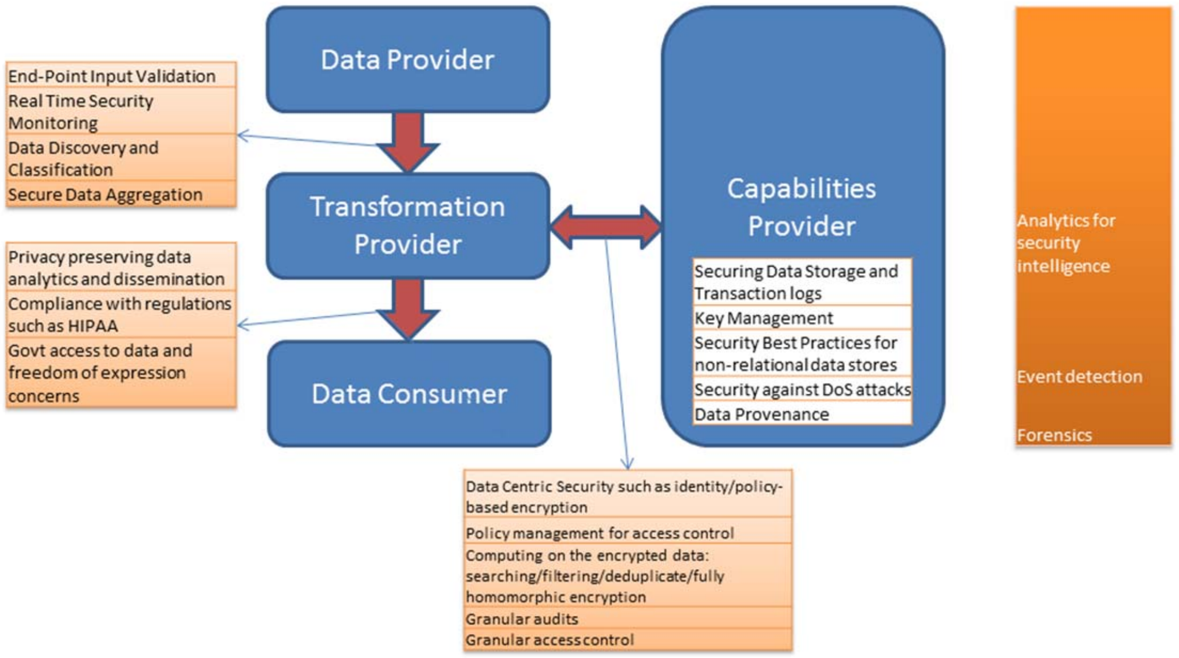

Data Provider
End-Point Input Validation
Real Time Security Monitoring
Data Discovery and Classification
Secure Data Aggregation
Transformation Provider
Privacy preserving data analytics and dissemination
Compliance with regulations such as HIPAA
Govt access to data and freedom of expression concerns
Data Consumer
Capabilities Provider
Securing Data Storage and Transaction logs
Key Management
Security Best Practices for non-relational data stores
Security against DoS attacks
Data Provenance
Analytics for security intelligence
Event detection
Forensics
Data Centric Security such as identity/policy-based encryption
Policy management for access control
Computing on the encrypted data: searching/filtering/deduplicate/fully homomorphic encryption
Granular audits
Granular access control

***Figure 2:***

# 6 Big Data Reference Architecture

<Content—a summary of Roadmap Volumes 5 and 6—is under development.>

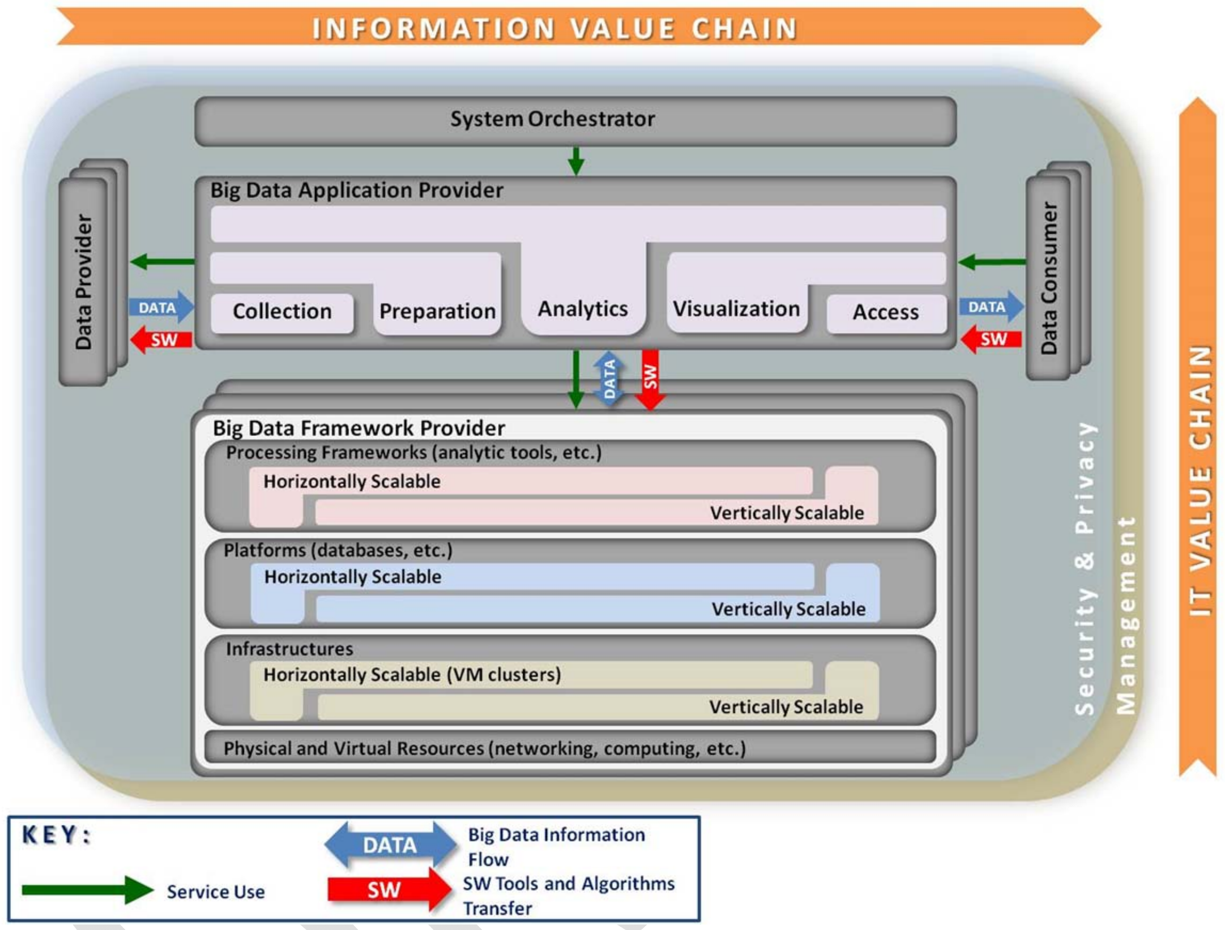


*Figure 3: The General Big Data Reference Architecture.*

# 7 Technology and Organizational Readiness and Features

<Introduction under development.>

## 7.1 Technology Readiness

The technological readiness for Big Data is a useful metric in assessing both the overall maturity of a technology across all implementers as well as the readiness of a technology for broad use within an organization. Technology readiness evaluates readiness types in a manner similar to that of technology readiness in Service-Oriented Architectures (SOA). However, the scale of readiness is adapted to better mimic the growth of open source technologies, notably those that follow models similar to the Apache Software Foundation (ASF). Figure 4 provides a superimposition of the readiness scale on a widely recognized "hype curve." This assures that organizations that have successfully evaluated and adopted aspects of SOA can apply similar processes to assessing and deploying Big Data technologies.

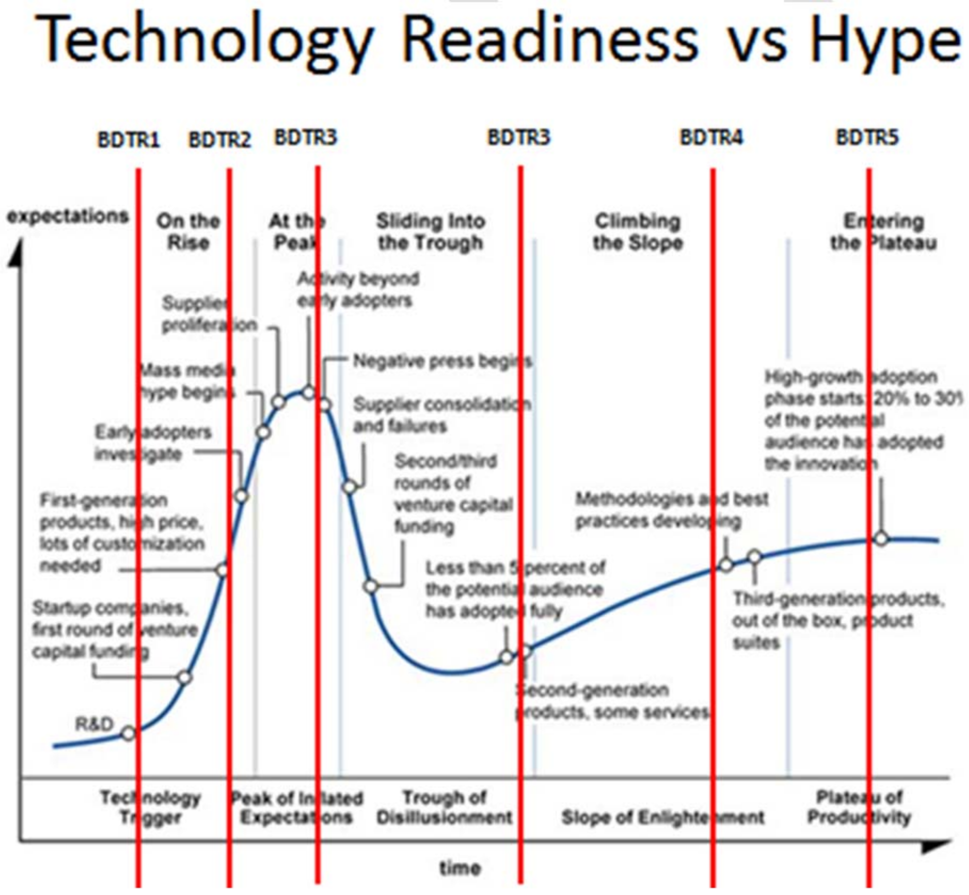


*Figure 5. Technology Readiness levels visualized along Gartner's "hype curve."*

### 7.1.1 Types of Readiness

<Introduction under development.>

- **Architecture**: Capabilities concerning the overall architecture of the technology and some parts of the underlying infrastructure
- **Deployment**: Capabilities concerning the architecture realization infrastructure deployment, and tools
- **Information**: Capabilities concerning information management: data models, message formats, master data management, etc.
- **Operations, Administration and Management**: Capabilities concerning post-deployment management and administration of the technology

### *7.1.2 Scale of Technological Readiness*

<Introduction under development.>

1. **Emerging**
    - Technology is largely still in research and development
    - Access is limited to the developers of the technology
    - Research is largely being conducted within academic or commercial laboratories
    - Scalability of the technology is not assessed
2. **Incubating**
    - Technology is functional outside laboratory environments
    - Builds may be unstable
    - Release cycles are rapid
    - Documentation is sparse or rapidly evolving
    - Scalability of the technology is demonstrated but not widely applied
3. **Reference Implementation**
    - One or more reference implementations are available
    - Reference implementations are usable at scale
    - The technology may have limited adoption outside of its core development community
    - Documentation is available and mainly accurate
4. **Emerging Adoption**
    - Wider adoption beyond the core community of developers
    - Proven in a range of applications and environments
    - Significant training and documentation is available
5. **Evolving**
    - Enhancement-specific implementations may be available
    - Tool suites are available to ease interaction with the technology
    - The technology competes with others for market share
6. **Standardized**
    - Draft standards are in place
    - Mature processes exist for implementation
    - Best practices are defined

## 7.2 Organizational Readiness and Adoption

In addition to looking at technology readiness, assessment of both the readiness of the organization and its level of adoption with respect to Big Data technologies are also essential. As with the domains and measures for the Technology Readiness scale described above, the following definitions are similar to those used for SOA.

### 7.2.1 Types of Readiness

The principal organizational readiness domains are defined as:

- **Business and Strategy:** Capabilities that provide organizational constructs necessary for Big Data initiatives to succeed. These include a clear and compelling business motivation for adopting Big Data technologies, expected benefits, funding models etc.
- **Governance:** The readiness of governance policies and processes to be applied to the technologies adopted as part of a Big Data initiative. Additionally, readiness of governance policies and processes for application to the data managed and operated on as part of a Big Data initiative.
- **Projects, Portfolios, and Services:** Readiness with respect to the planning and implementation of Big Data efforts. Readiness extends to quality and integration of data, as well as readiness for planning and usage of Big Data technology solutions.
- **Organization:** Competence and skills development within an organization regarding the use and management of Big Data technologies. This includes, but is not limited to, readiness within IT departments (e.g., service delivery, security, and infrastructure) and analyst groups (e.g. methodologies, integration strategies, etc.).

### 7.2.2 Scale of Organizational Readiness

<Introduction under development.>

1. **No Big Data**
    - No awareness or efforts around Big Data exist in the organization
2. **Ad Hoc**
    - Awareness of Big Data exists
    - Some groups are building solutions
    - No Big Data plan is being followed
3. **Opportunistic**
    - An approach to building Big Data solutions is being determined
    - The approach is opportunistically applied, but is not widely accepted or adopted within the organization
4. **Systematic**
    - The organizational approach to Big Data has been reviewed and accepted by multiple affected parties.
    - The approach is repeatable throughout the organization and nearly-always followed.
5. **Managed**
    - Metrics have been defined and are routinely collected for Big Data projects
    - Defined metrics are routinely assessed and provide insight into the effectiveness of Big Data projects
6. **Optimized**
    - Metrics are always gathered and assessed to incrementally improve Big Data capabilities within the organization.

- Guidelines and assets are maintained to ensure relevancy and correctness

### 7.2.3 Scale of Organizational Adoption

<Introduction under development.>

1. **No Adoption**
   - No current adoption of Big Data technologies within the organization
2. **Project**
   - Individual projects implement Big Data technologies as they are appropriate
3. **Program**
   - A small group of projects share an implementation of Big Data technologies
   - The group of projects share a single management structure and are smaller than a business unit
4. **Divisional**
   - Big Data technologies are implemented consistently across a business unit
5. **Cross-Divisional**
   - Big Data technologies are consistently implemented by multiple divisions with a common approach
   - Big Data technologies across divisions are at an organizational readiness level of Systematic or higher
6. **Enterprise**
   - Big Data technologies are implemented consistently across the enterprise
   - Organizational readiness is at level of Systematic or higher

## 7.3 Features

The ability for technical and business stakeholders to view the current and future state of features enables them to better make decisions in using Big Data. The three central components—technology, problems, and end —are in constant change, independently and interactively. When one moves, the others are affected and leaders must respond accordingly. The objective of this Roadmap is to outline the future of Big Data as well as possible and provide essential decision-making information. Put another way, it aims to be the best available snapshot in time of this moving target.

In this Roadmap, the features that outline the current and future state of Big Data fall into four primary categories: 1) data services, 2) usage services, 3) capabilities, and, 4) vertical orchestrator. These four categories mirror the Big Data reference architecture and taxonomy that summarized in this volume and described in detail in companion volumes in this set. Within these categories are nine top features, as shown in Table 1.

NIST Working Group arrived at these four categories and nine features by rationalizing of the Big Data landscape. The four categories were arrived at by aligning to the reference architecture. The nine features were gleaned from the reference architecture, requirements (use cases), (architectural) capabilities, actors, taxonomy, and [Big Data] readiness material that are part of the roadmap set. Figure 6 displays the inputs and relationships between these elements and this Roadmap.

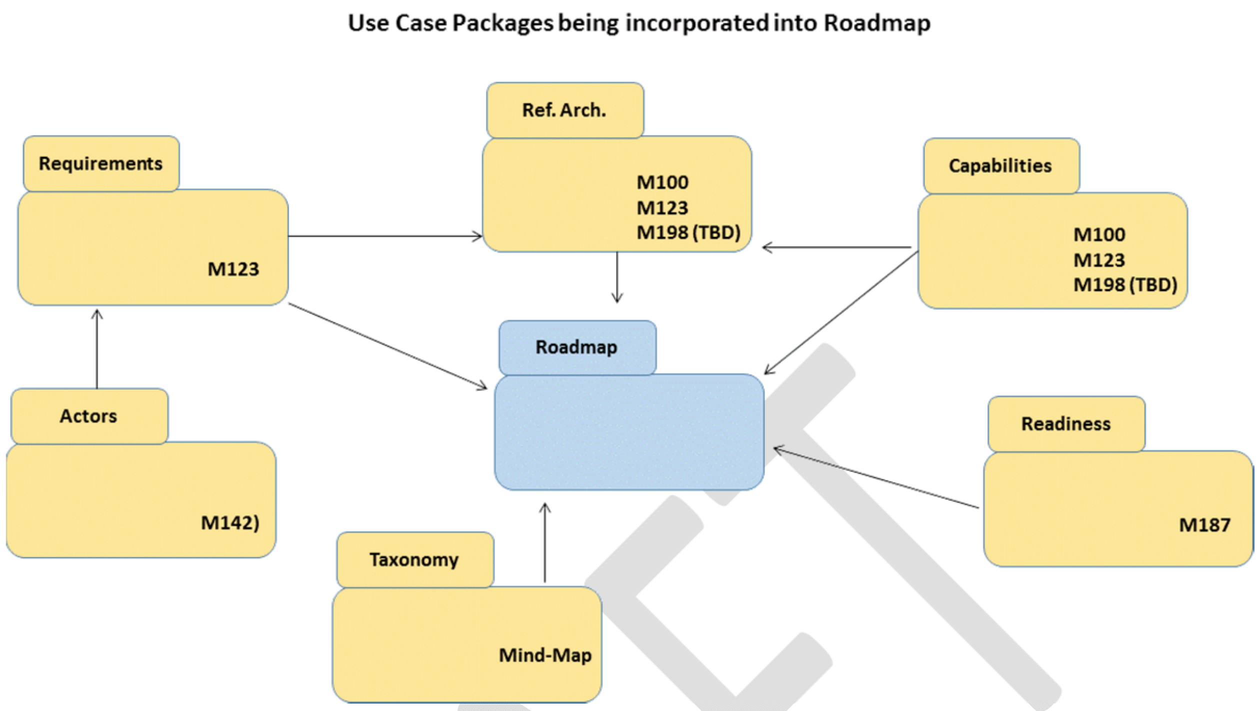


*Figure 7. Incorporation of Use Cases in This Roadmap.*

The centricity of the end-user is critical, so the Roadmap has adopted four actor groups from the roadmap set, as follows and shown in Figure 8 below: 1) System Controller; 2) Data Transformation; 3) Data Consumer; and 4) Vertical Orchestrator (see diagram below). Similar to Unified Modeling Language (UML) standards, the actors are abstract and can be either individuals or systems. Actors can also fulfill more than one role.

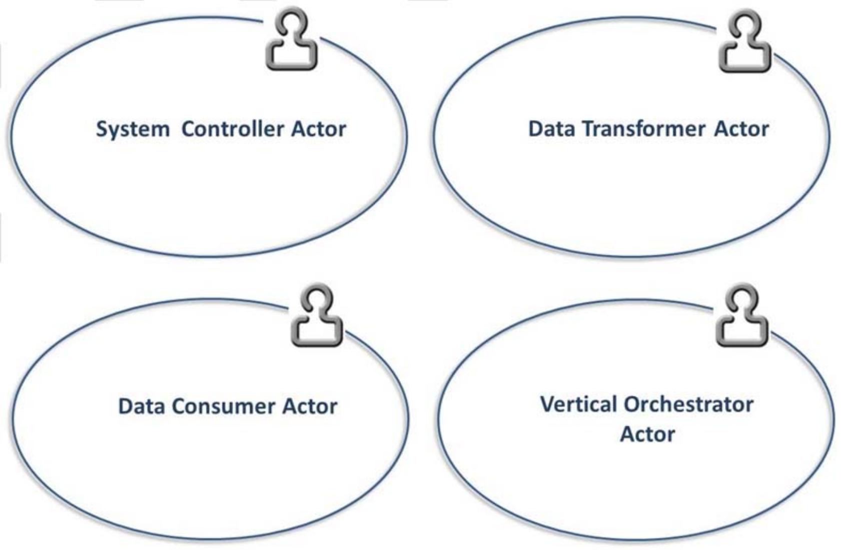


*Figure 9. Four Actor Groups in Big Data.*

Table 1 provides value statements for each of the Big Data features, and includes mapping of each feature to technology and organizational readiness.

*Table 1. Value Statements and Readiness Mapping for the Nine Big Data Features.*

| Feature | Value Statement | Roles | Readiness | RA Mapping |
|---|---|---|---|---|
| **1. Storage Framework** | Storage Framework defines how Big Data is logically organized, distributed, and stored. The volume and velocity of Big Data frequently means that traditional (file systems, RDBMS) solutions will not hold up to one or both of these attributes. | TBD | TBD | Capabilities |
| **2. Processing Framework** | Processing Frameworks defines how data is operated on in the Big Data environment. The volume or velocity of Big Data often means that analysis of the data requires more resources (memory, CPU cycles) than are available on a single compute node and that the processing of the data must be distributed and coordinated across many nodes. | TBD | TBD | Capabilities |
| **3. Resource Managers Framework** | Because many Big Data storage and processing Frameworks are distributed and no single storage and processing solution may meet the needs of the end user, resource management solutions are required to manage and allocate resources across disparate frameworks. | TBD | TBD | Capabilities |
| **4. Infrastructure Architecture** | Big Data requires the ability to operate with sufficient network and infrastructure backbone. For Big Data to deploy, it is critical that the Infrastructure Framework has been right-sized. | TBD | TBD | Capabilities |
| **5. Information Architecture** | Prior to any Big Data decision, the data itself needs to be reviewed for its informational value. | TBD | TBD | Data Services |
| **6. Standards Integration Framework** | Integration with appropriate standards (de jure, consortia, reference implementation, open source implementation, etc.) can assist both in cross-product | TBD | TBD | Data Services |

| Feature | Value Statement | Roles | Readiness | RA Mapping |
|---|---|---|---|---|
| | integration and cross product knowledge. Identifying which standards efforts address architectural requirements and which requirements are not currently being addressed provides input for future standards efforts. | | | |
| **7. Applications Framework** | The building blocks of applications are data. The Application Lifecycle Management needs to take into consideration how applications will interact with a Big Data solution. | TBD | TBD | Capabilities |
| **8. Business Operations** | Big Data is more than just technology, but also a cultural and organizational transformation. Business Operations need to be able to strategize, deploy, adopt, and, operate Big Data solutions. | TBD | TBD | Vertical Orchestrator |
| **9. Business Intelligence** | The presenting of data into information, intelligence, and, insight is typically the end value of Big Data. | TBD | TBD | [Will be moved under ‘Applications’] |

The top features of the Roadmap can also be viewed visually. Two examples illustrating relevant decision-making criteria against the Roadmap categories and features are provided below. These diagrams can help enable decision-makers to see "around the corner" in their Big Data discussions.

The diagram in Figure 4 groups Roadmap features into the four categories as "swim lanes”, with examples of the features spanning readiness. The mini-pie charts show the level of value, security, or privacy.

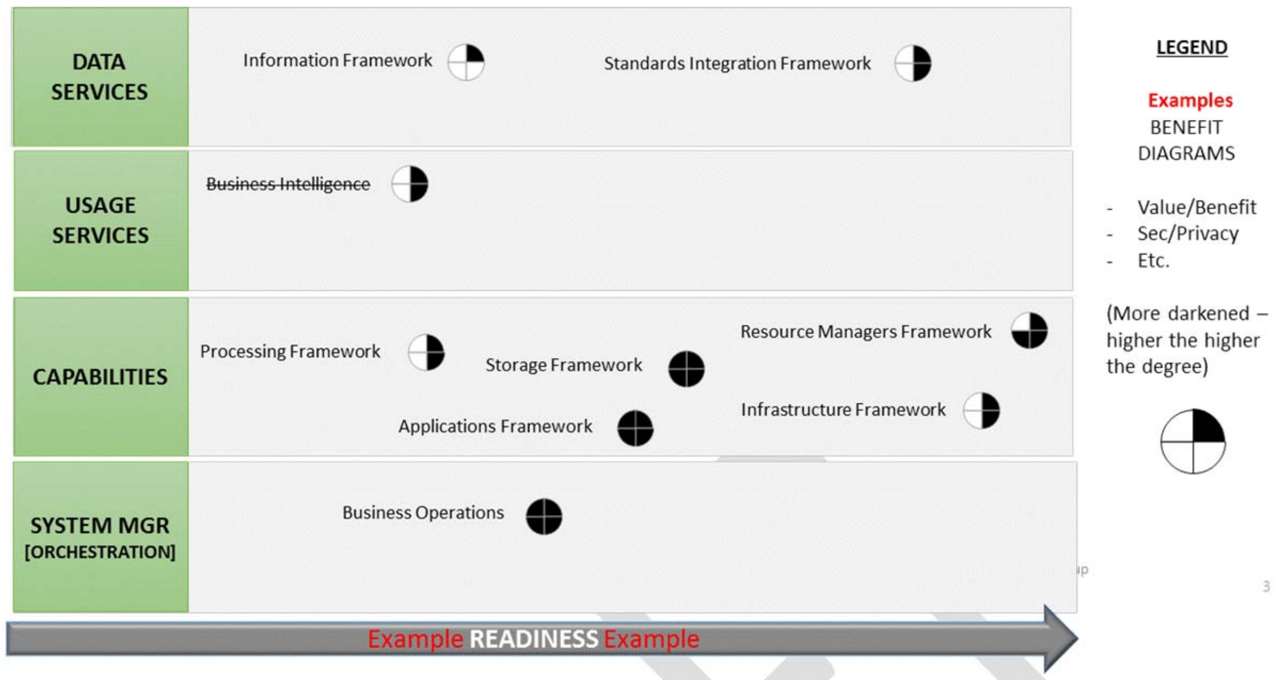


*Figure 10. Roadmap Features across Categories.*

Figure 11 is a diagram that displays the Roadmap visually as a scatter diagram with an X-axis (life of the data) and a Y-axis (level of readiness). Roadmap features are color-coded to the four categories. Their placement in the diagram indicates where they sit in terms of the longitudinal use of the data—transactional to analytical—and the level of their organizational and architectural readiness for adoption.

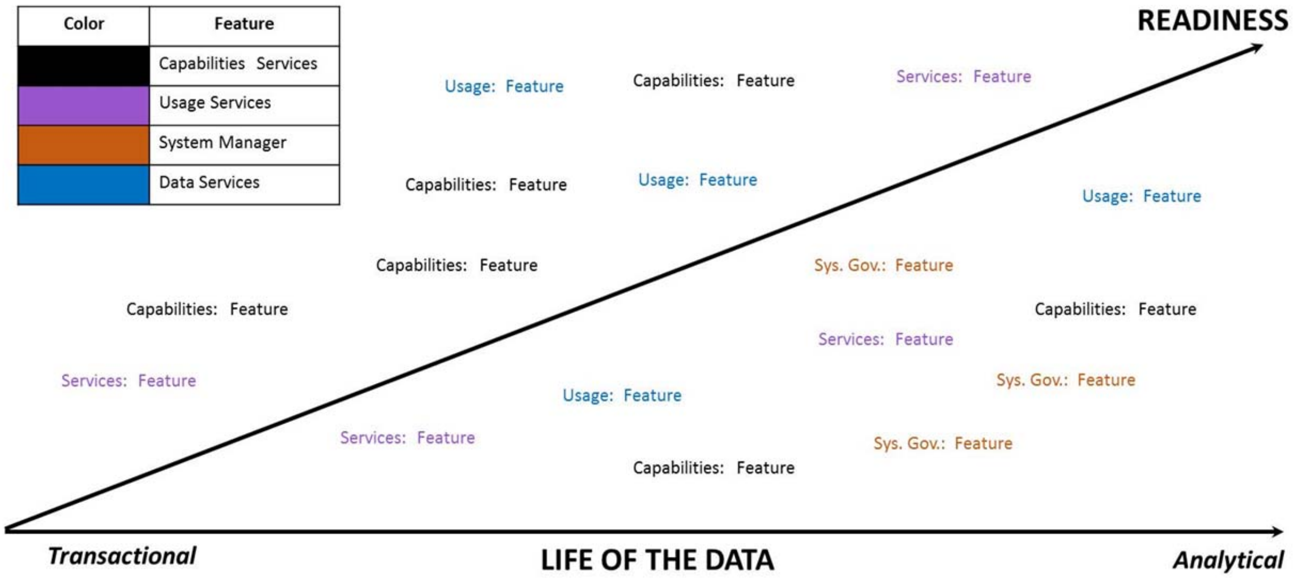


*Figure 12. Roadmap Characteristics as Scatter Diagram.*

### 7.3.1 Feature 1: Storage Framework

Storage frameworks have and continue to undergo dramatic changes as the need to capture, store, process, and archive ever-larger amounts of data. The nature of this increase covers both the total volume of “data artifacts” (e.g., 9100 Tweets per second[5]) to the extreme size of individual artifacts (e.g., 110 megapixel images from the Vigilant Stare Wide Area Persistent Surveillance (WAPS) Platform[6]). In order to support the processing of data across this continuum, unique and novel approaches are frequently required to support not just the storage but also the indexing for access/processing and the preservation/backup and transfer of the data.

Typically, storage frameworks consist of two interdependent aspects: the physical organization of the data on the media, and the logical organization of the data within the physical layout. This Roadmap specifically focuses on persistent storage approaches, though it should be noted that most of these approaches could be implemented in non-persistent storage (e.g. RAM disk). It does not delve into the physical media itself. While the primary media are traditional spinning magnetic disks and Solid State Disks built on flash memory technology today, new storage media technologies such as holographic, quantum, and nano-bubble are under development and maturing rapidly. In addition, the density of current prevalent data storage and emerging technologies continues to increase roughly along Moore’s Law. From a Big Data perspective, this generally means that more data can be stored in a smaller footprint and, in general (especially where mechanical processes like disk head seeks are involved), access times will continue to decrease and throughput to increase. As these new media types mature and their performance characteristics are better understood, they should be able to serve as drop-in replacements for existing storage media.

#### 7.3.1.1 Physical Storage Frameworks

The physical organization of storage generally follows the continuum from local to distributed as shown in Figure 13 below. Associated technology readiness is indicated by the number in the circle.

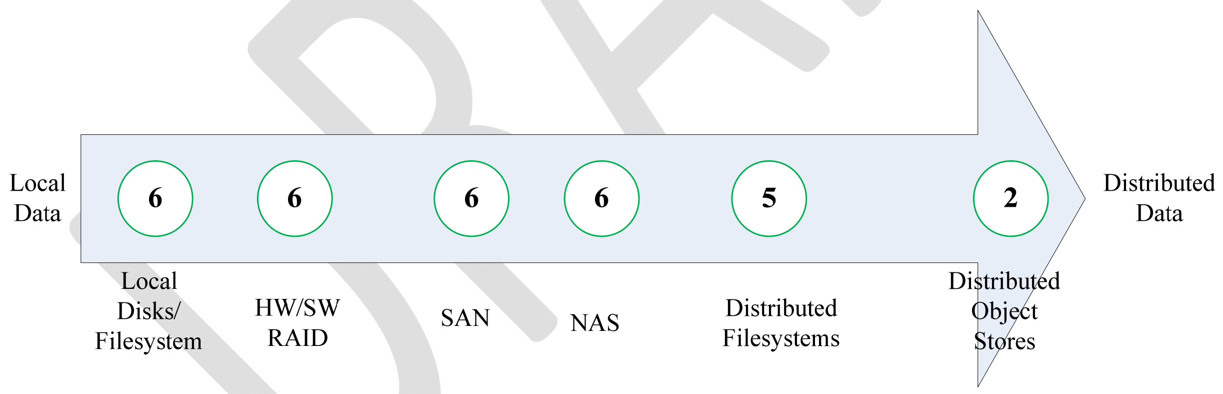


*Figure 14. The Physical Data Storage Continuum.*

There are two aspects of these technologies that directly influence their suitability for Big Data solutions. First, there is capacity (dealing with the volume of the data). Local disks/file systems are specifically limited by the size of the available media. Hardware vs. software (HW/SW) Redundant Array of Independent Disks (RAID) solutions (in this case local to a processing node) help that scaling by allowing multiple pieces of media to be treated as a single device. However, that approach is limited by the physical dimension of the media and the number of devices the node can accept. Storage area networks (SAN) and network attached storage (NAS) implementations (often known as shared disk solutions) remove that limit by consolidating storage into a storage specific device. However, they start to run into the second influencing factor that is transfer bandwidth. While both network and input/output (I/O) interfaces are getting faster and many implementations support multiple transfer channels I/O bandwidth can be a limiting factor. In addition, despite the redundancies provided by RAID, hot spares (fallback mechanism), multiple power supplies, and multiple controllers, these boxes can often become I/O bottlenecks or single points of failure in an enterprise. Distributed file systems (DFS)—also known as cluster file systems—seek to overcome these issues by combining I/O throughput through multiple devices (spindles) on each node with redundancy and failover by mirroring/replicating data at the block level across multiple nodes. This specifically is designed to allow the use of heterogeneous commodity hardware across the Big Data cluster. Thus, if a single drive or an entire node should fail, no data is lost because it is replicated on other nodes and throughput is only minimally impacted as processing can be moved to other nodes. In addition, replication allows for high levels of concurrency for reading data and for initial writes. Updates and transaction style changes tend to be an issue for many distributed file systems because time delays in creating replicated blocks will create consistency issues (e.g. a block is changed but another node reads the old data before it is replicated).

Unlike the other technologies described here which implement a traditional file system approach, Distributed object stores (sometimes called global object stores) present a flat name space with a globally unique identifier (GUID) for any given chunk of data. Data in the store is located generally through a query against a metadata catalog that returns the associated GUID(s). The underlying implementation of the software generally knows from the GUID where that particular chunk of data is stored. These object stores are being developed and marketed for storage of very large data objects from complete data sets to large individual objects (i.e., high-resolution images in the tens of gigabytes size range). The biggest limitation of these stores for Big Data problems tends to be network throughput, since many require the object to be accessed in total. However, trends point to the concept of being able to send the computation/application to the data versus needing to bring the data to the application in the future.

From a maturity perspective, the two key areas where distributed file systems would be expected to improve are on random write I/O performance and consistency and the generation of standards at least at the level of the Internet Engineering Task Force (EIETF) Request for Comments (RFC) available today for network file systems (NFS). Distributed object stores while currently available and operational (e.g., Amazon S3) and part of the roadmap for large organizations such as the National Geospatial Intelligence Agency (NGA), currently are essentially specialized/proprietary implementations. To become prevalent within Big Data ecosystems, there will need to be some level of interoperability available (through standardized application programming interfaces [APIs]), standards-based approaches for data discovery, and—probably most important—standards-based approaches that allow the application to be transferred over the grid and run locally to the data versus the need for the data to be transferred to the application.

#### 7.3.1.2 Logical Data Distribution

In most aspects, the logical distribution/organization of data in Big Data storage frameworks mirrors what is common for most legacy systems. Figure 15 below shows a brief overview of data organizations approaches for Big Data.

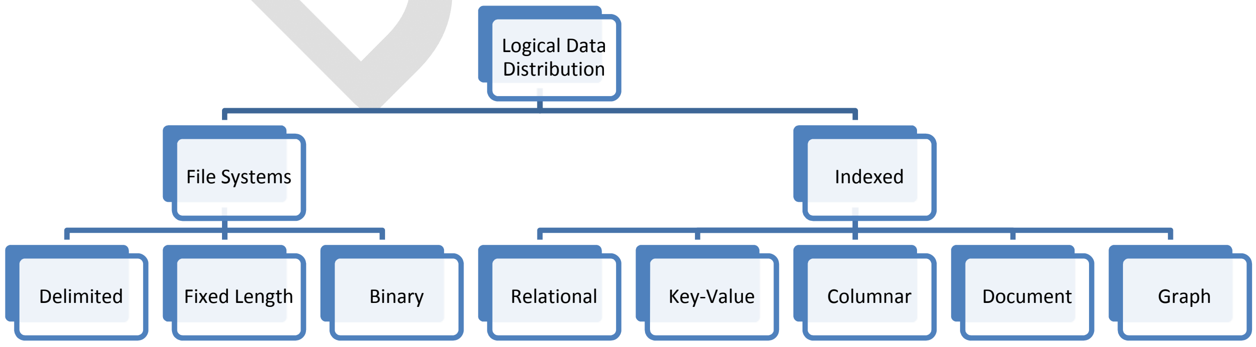


***Figure 16. Logical Data Distribution Framework.***

As mentioned, many Big Data logical storage organizations leverage the common file system concept (where chunks of data are organized into a hierarchical namespace of directories) as their base and then implement various indexing methods within the individual files. This allows many of these approaches to be run both on simple local storage file systems for testing purposes or on fully distributed file systems for scale.

### File Systems

Many Big Data processing frameworks and applications are content to access their data directly from an underlying file system. In almost all cases, the file systems implement some level of the Portable Operating System Interface (POSIX) standards for permissions and the associated file operations. This allows other higher-level frameworks for indexing or processing operate to be relatively transparent, whether the underlying file system is local or fully distributed. Within files systems, there is nothing new or novel in the storage of data. It can be text or binary data, fixed length records, or some sort of delimited structure (e.g., comma-separated values, Extensible Markup Language [XML]). Several of these file system implementations also support data compression and encryption at various levels. The one major caveat to this is that for distributed block-based file systems, the compression/encryption must be splittable and allow any given block to be decompressed/decrypted out of sequence and without access to the other blocks. For record-oriented storage (delimited or fixed length), this generally is not a problem unless individual records can exceed a block size. Some distributed file system implementations provide compression at the volume or directory level and implement it below the logical block level (e.g., when a block is read from the file system it is decompressed/decrypted before being returned). Because of its simplicity, familiarity, and portability, delimited files are frequently the default storage format in many Big Data implementations. The trade-off for this is I/O efficiency. While individual blocks in a distributed file system might be accessed in parallel, each block still needs to be read in sequence. In the case of a delimited file, if you are interested in only the last field of certain records with maybe hundreds of fields, you will have a lot of wasted I/O and processing bandwidth.

Binary formats tend to be application- or implementation-specific. While they can offer much more efficient access both due to smaller data sizes (integers are 2-4 bytes in binary while they are 1byte per digit in ASCII), they offer limited portability between different implementations. At least one popular distributed file system provides its own standard binary format that at least allows data to be portable between multiple applications without additional software. That said, the bulk of the indexed data organization approaches discussed below leverage binary formats for efficiency.

### Indexed Storage Organization

The very nature of Big Data (volume and velocity primarily) drives requirements to some form of indexing structure. The volume requires that specific elements of data can be located quickly without scanning across the entire dataset. The velocity also requires that data can be located quickly, either for matching (e.g. does any incoming data match something in my existing data set) or to know where to write/update new data.

The choice of a particular indexing method or methods depends mostly on your data and the nature of the application you are trying to implement. For example, graph data (vertexes, edges, and properties) can be easily represented in flat text files as vertex, edge pairs, edge, vertex, vertex triples, or vertex, edge list records. However, processing this data efficiently would require potentially loading the entire data set into memory or being able to distribute the application and data set across multiple nodes so a portion of the graph is in memory on each node. That would then require the nodes to communicate when graph sections have vertices that connect with vertices on other processing nodes. This is perfectly acceptable for some graph applications, such as shortest path, especially when the graph is static. Moreover, some graph processing frameworks operate using this exact model. However, if the graph is dynamic or you need to quickly search or match to a portion of the graph, then this approach becomes infeasible for large-scale graphs requiring a specialized graph storage framework.

The indexing approaches described below tend to be classified by the features typically provided in the implementation, i.e., the complexity of the data structures that can be stored, how well they can process links between data, and how easily they support multiple access patterns as shown in Figure 17 below. Since any of these features can be implemented in custom application code, the values portrayed represent approximant norms. For example, key-value stores work well for data that is only accessed through a single key, whose values can be expressed in a single flat structure, and where multiple records do not need to be related. Document stores can support very complex structures of arbitrary width and tend to be indexed for access via multiple document properties but do not tend to support inter-record relationships well. In reality, the specific implementations for each storage approach vary enough so that all of the values for the features represented are really ranges, e.g., relational data storage implementations are supporting increasingly complex data structures and there is work going on to add more flexible access patterns natively in BigTable columnar implementations. Within Big Data, the performance of each of these features tends to drive the scalability of that approach depending on the problem being solved. For example, if the problem is to locate a single piece of data for a unique key, then key-value stores will scale really well. On the other hand, if the problem requires general navigation of the relationships between multiple data records, then a graph storage model will likely provide the best performance.

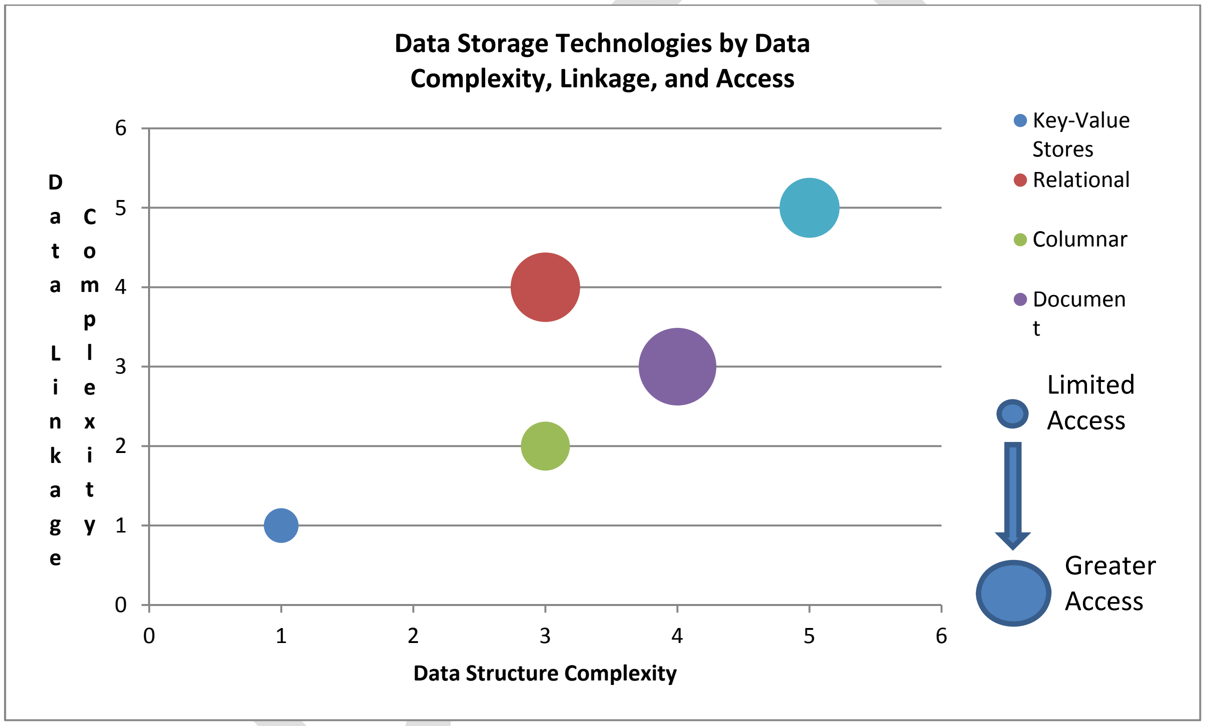


*Figure 18. Big Data Indexing Approaches.*

The following paragraphs describe several indexing approaches for storing Big Data and the advantages and issues commonly found for each one.

### *Relational Storage Models*

Relational storage models are perhaps the most familiar, as the basic concept has existed since the 1950s and the Structured Query Language (SQL) is a mature standard for manipulating (i.e., searching, inserting, updating, or deleting) relational data. In this model, data is stored as rows, with each field representing a column in a table based on the logical data organization. The problem with relational storage models and Big Data is the join between one or more tables. While the size of two or more tables of data individually might be small, the join (or relational matches) between those tables will generate exponentially more records. The appeal of this model for organizations just adopting Big Data is its familiarity. The pitfalls are its limitations and performance expectations, plus the tendency to adopt standard relational database management system (RDBMS) practices such as high normalization, detailed, and specific indices.

Big data implementations of relational storage models are relatively mature and have been adopted by a number of organizations. They are also maturing very rapidly with new implementations focusing on improved response time. Many Big Data implementations take a brute force approach to scaling relational queries. Essentially, queries are broken into stages and processing of the input tables is distributed across multiple nodes (often as a MapReduce job). The actual storage of the data can be flat files (delimited or fixed length) where each record/line in the file represents a row in a table. Increasingly, these implementations are adopting binary storage formats optimized for distributed file systems. These formats often use block-level indices and column-oriented organization of the data to allow individual fields to be accessed in records without needing to read the entire record. Despite this, most Big Data relational storage models are still "batch-oriented" systems designed for very complex queries that generate very large intermediate cross-product matrices from joins, so even the simplest query can require tens of seconds to complete. There is significant work going on and emerging implementations that are seeking to provide a more interactive response and interface.

Early implementations only provided limited data types and little or no support for indices. However, most current implementations have support for complex data structures and basic indices. However, while the query planners/optimizers for most modern RDBMS systems are very mature and implement cost-based optimization through statistics on the data the query planners/optimizers in many Big Data implementations remain fairly simple and rule-based in nature. While generally acceptable for batch-oriented systems (since the scale of processing the Big Data in general can be orders of magnitude more an impact), any attempt to provide interactive response will need very advanced optimizations so that (at least for queries) only the most likely data to be returned is actually searched. This leads to the single most serious drawback with many of these implementations. Since distributed processing and storage are essential for achieving scalability, these implementations are directly limited by the Consistency, Availability, and Partition Tolerance (CAP) theorem. Many in fact provide what is generally referred to a t-eventual consistency which means that barring any updates to a piece of data all nodes in the distributed system will eventually return the most recent value. This level of consistency is typically fine for Data Warehousing applications where data is infrequently updated and updates are generally done in bulk. However, transaction oriented databases typically require some level of Atomicity, Consistency, Isolation, Durability (ACID) compliance to insure that all transactions are handled reliably and conflicts are resolved in a consistent manner. There are a number of both industry and open source initiatives looking to bring this type of capability to Big Data relational storage frameworks. One approach is to essentially layer a traditional RDBMS on top of an existing distributed file system implementation. While vendors claim that this approach means that the overall technology is mature, a great deal of research and implementation experience is needed before the complete performance characteristics of these implementations are known.

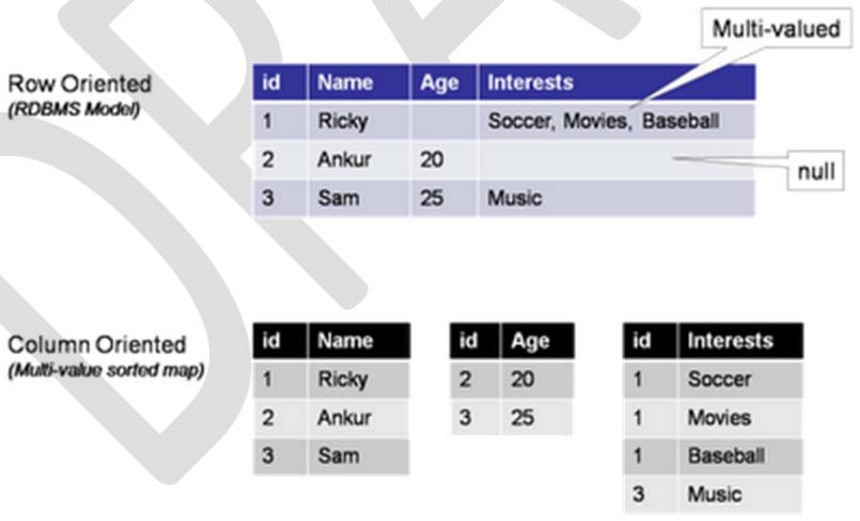
#### *Key-Value Storage Models*

Key-value stores are one of the oldest and most mature data indexing models. In fact, the principles of key value stores underpin all the other storage and indexing models. From a Big Data perspective, these stores effectively represent random access memory models. While the data stored in the values can be arbitrarily complex in structure, all the handling of that complexity must be provided by the application with the storage implementation often providing back just a pointer to a block of data. Key-value stores also tend to work best for one-to-one relationships (e.g., each key relates to a single value), but can also be effective for keys mapping to lists of homogeneous values. When keys map multiple values of heterogeneous types/structures or when values from one key need to be joined against values for a different or the same key, then custom application logic is required. It is the requirement for this custom logic that often prevents key-value stores from scaling effectively for certain problems. However, depending on the problem, certain processing architectures can make effective use of distributed key-value stores. Key-value stores generally deal well with updates when the mapping is one-to-one and the size/length of the value data does not change. The ability of key-value stores to handle inserts is generally dependent on the underlying implementation. Key-value stores also usually require significant effort (either manual or computational) to deal with changes to the underlying data structure of the values.

Distributed key-value stores are the most frequent implementation utilized in Big Data applications. One problem that must always be addressed (but is not unique to key-value implementations) is the distribution of keys across the space of possible key values. Specifically, keys must be chosen carefully to avoid skew in the distribution of the data across the cluster. When data is heavily skewed to a small range, it can result in computation hot spots across the cluster if the implementation is attempting to optimize data locality. If the data is dynamic (e.g., new keys being added), then it is likely that at some point the data will require rebalancing across the cluster. Non-locality optimizing implementations employ various sorts of hashing, random, or round robin approaches to data distribution and do not tend to suffer from skew and hot spots. However, they perform poorly on problems requiring aggregation across the data set.

### *Columnar Storage Models*

Much of the hype associated with Big Data came with the publication of the BigTable paper by Google in 2006[7], but column-oriented storage models like BigTable are not new and have been stalwarts of the data-warehousing domain for many years. Unlike traditional relational data that store data by rows of related values, columnar stores organize data in groups of like values. The difference is subtle, but in relational databases, an entire group of columns are tied to some primary key (frequently one or more of the columns) to create a record. In columnar, the value of every column is a key and like column values point to the associated rows. The simplest instance of a columnar store is little more than a key-value store with the key and value roles reversed. In many ways, columnar data stores look very similar to indices in relational databases. Figure 19 below shows the basic differences between row- and column-oriented stores.


Multi-valued
null

Row Oriented
*(RDBMS Model)*

| id | Name | Age | Interests |
|---|---|---|---|
| 1 | Ricky | | Soccer, Movies, Baseball |
| 2 | Ankur | 20 | |
| 3 | Sam | 25 | Music |

Column Oriented
*(Multi-value sorted map)*

| id | Name |
|---|---|
| 1 | Ricky |
| 2 | Ankur |
| 3 | Sam |

| id | Age |
|---|---|
| 2 | 20 |
| 3 | 25 |

| id | Interests |
|---|---|
| 1 | Soccer |
| 1 | Movies |
| 1 | Baseball |
| 3 | Music |

***Figure 20. Illustration of the Differences between Row-and Column-oriented stores.***

In addition, implementations of columnar stores that follow the Google BigTable model introduce another level of segmentation beyond the table, row, and column relational models. That is called the column family. In those implementations, rows have a fixed set of column families but within a column family, each row can have a variable set of columns. This is illustrated in Figure 21 below.

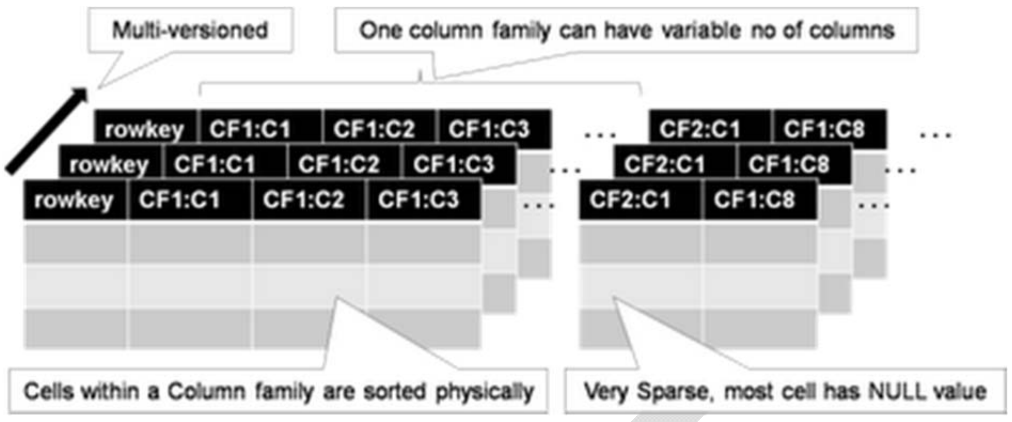

Multi-versioned
One column family can have variable no of columns
rowkey
CF1:C1
CF1:C2
CF1:C3
CF2:C1
CF1:C8
Cells within a Column family are sorted physically
Very Sparse, most cell has NULL value

***Figure 22. Illustration of a Column Family.***

The key distinction in the implementation of columnar stores over relational stores is that data is denormalized for column stores and, while for relational stores every record contains some value (perhaps null) for each column, in columnar stores the column is only present if there is data for one or more rows. This is why many column-oriented stores are referred to as “sparse storage models.” Data for each column family is physically stored together on disk sorted by row, column name, and timestamp. The timestamp is there because the BigTable model also includes the concept of versioning. Every, RowKey, Column Family, Column triple is stored with either a system-generated or user-provided timestamp. This allows users to quickly retrieve the most recent value for a column (the default), the specific value for a column by timestamp, or all values for a column. The last is most useful because it permits very rapid temporal analysis on data in a column.

Because data for a given column is stored together, two key benefits are achieved. First, aggregation of the data in that column requires only the values for that column to be read. Conversely, in a relational system the entire row (at least up to the column) needs to be read (which if the row is long and the column at the end it could be lots of data). Second, updates to a single column do not require the data for the rest of the row to be read/written. Also, because all the data in a column is uniform, data can be compressed much more efficiently. Often, only a single copy of the value for a column is stored followed by the RowKeys where that value exists. And, while deletes of an entire column is very efficient, deletes of an entire record are extremely expensive. This is why historically column oriented stores have been applied to online analytical processing (OLAP) style applications while relational stores were applied to online transaction processing (OLTP) requirements.

***Column Storage and Security***

*Recently, security has been a major focus of existing column implementations, primarily due to the release by the National Security Agency (NSA) of its BigTable implementation to the open source community. A key advantage of the NSA implementation and other recently announced implementations is the availability of security controls at the individual cell level. With these implementations, a given user might have access to only certain cells in a group based on the value of those or other cells.*

There are several very mature distributed column-oriented implementations available today from both open source groups and commercial foundations. These have been implemented and are operational across a wide range of businesses and government organizations. Emerging are hybrid capabilities that implement relational access methods (e.g. SQL) on top of BigTable/Columnar storage models. Also, relational implementations are adopting columnar-oriented physical storage models to provide more efficient access for Big Data OLAP-like aggregations and analytics.

#### *Document*

Document storage approaches have been around for some time and popularized by the need to quickly search large amounts of unstructured data. Modern document stores have evolved to include extensive search and indexing capabilities for structured data and metadata, and they are often referred to as "semi-structured data stores." Within a document-oriented data store, each "document" encapsulates and encodes the metadata, fields, and any other representations of that record. While somewhat analogous to a row in a relational table, one reason document stores evolved and have gained in popularity is that most implementations do not enforce a fixed or constant schema. While best practices hold that groups of documents should be logically related and contain similar data, there is no requirement that they be alike or that any two documents even contain the same fields. That is one reason that document stores are frequently popular for data sets that have sparsely populated fields, since there is far less overhead normally than traditional RDBMS systems where null value columns in records are actually stored. Groups of documents within these types of stores are generally referred to as collections and, like key-value stores, some sort of unique key references each document.

In modern implementations, documents can be built of arbitrarily nested structures and can include variable length arrays and in some cases executable scripts/code (which has significant security and privacy implications). Most document-store implementations also support additional indices on other fields or properties within each document, with many implementing specialized index types for sparse data, geospatial data, and text.

When modeling data into document-stores, the preferred approach is to denormalize the data as much as possible and embed all one-to-one and most one-to-many relationships within a single document. This allows for updates to documents to be atomic operations that keep referential integrity between the documents. The most common case where references between documents should be use is when there are data elements that occur frequently across sets of documents and whose relationship to those documents is static.

In the Big Data realm, document stores scale horizontally using partitioning or sharding to distribute portions of the collection across multiple nodes. This partitioning can be round robin-based, assuring an even distribution of data, or content/key based, so that data locality is maintained for similar data. Depending on the application required, as with any database the choice of partitioning key can have significant impacts on performance, especially where aggregation functions are concerned.

There are no standard query languages for document store implementations with most using a language derived from their internal document representation (e.g., JavaScript Object Notation [JSON], XML).

***References Between Documents***

*As an example, the publisher of a given book edition does not change and there are far fewer publishers than there are books. It would not make sense to embed all the publisher information into each book document. Rather the book document would contain a reference to the unique key for the publisher. Since for that edition of the book the reference will never change and so there is no danger of loss of referential integrity. Thus, information about the publisher (e.g., address) can be updated in a single atomic operation the same as the book. Where this information embedded, it would need to be updated in every book document with that publisher.*

#### *Graph*

Graph stores represent data as a series of nodes, edges, and properties on those. Analytics against graph stores include very basic shortest path and page ranking to entity disambiguation and graph matching. While social networking sites like Facebook and LinkedIn have driven the visibility of and evolution of graph stores (and processing as discussed below), graph stores have been a critical part of many problem domains from military intelligence and counter terrorism to route planning/navigation and the semantic web for years.

Graph databases typically store two types of objects nodes and relationships as show in Figure 23 below. Nodes represents objects in the problem domain that are being analyzed be they people, places, organizations, accounts, etc. Relationships describe those objects in the domain relate to each other. Relationships can be nondirectional or bidirectional, but are typically expressed as unidirectional in order to provide more richness and expressiveness to the relationships. Within graphs, relationships are not always equal or have the same strength. Nodes and relationships can have properties or attributes.

***Relationships in Graph Data***

*As an example, between two people nodes where they are father and son, there would be two relationships: 1) "is father of" (going from the father node to the son node), and 2) "is son of" (going from the son node to the father node). Properties or attributes of nodes and relationships are typically descriptive data about the element. For people it might be name, birth date, etc. For locations, it might be an address or geospatial coordinate. For a relationship like a phone call, it could be the date, time of the call, and the duration of the call. In terms of properties, relationships often have one or more weight, cost, or confidence attributes. A strong relationship between people might have a high weight because they have known each other for years and communicate every day. A relationship where two people just met would have a low weight.*

The distance between nodes (be it a physical distance or a difficulty) is often expressed as a cost attribute on a relation in order to allow computation of true shortest paths across a graph. In military intelligence applications, relationships between nodes in a terrorist or command and control network might only be suspected or have not been completely verified, so those relationships would have confidence attributes. Properties on nodes may also have confidence factors associated with them, although in those cases the property can be decomposed into its own node and tied with a relationship.

Graph storage approaches can actually be viewed as a specialized implementation of a document storage scheme with two types of documents (nodes and relationships). In addition, one of the most critical elements in analyzing graph data is locating the node or edge in the graph where you want to begin the analysis. To accomplish this, most graph databases implement indices on the node or edge properties. Unlike relational and other data storage approaches, most graph databases tend to use artificial/pseudo keys or guides to uniquely identify nodes and edges. This allows attributes/properties to be easily changed due to both actual changes in the data (e.g., someone changed their name) or as more information is found out (e.g., we get a better location for some item or event) without needing to change the pointers two/from relationships.

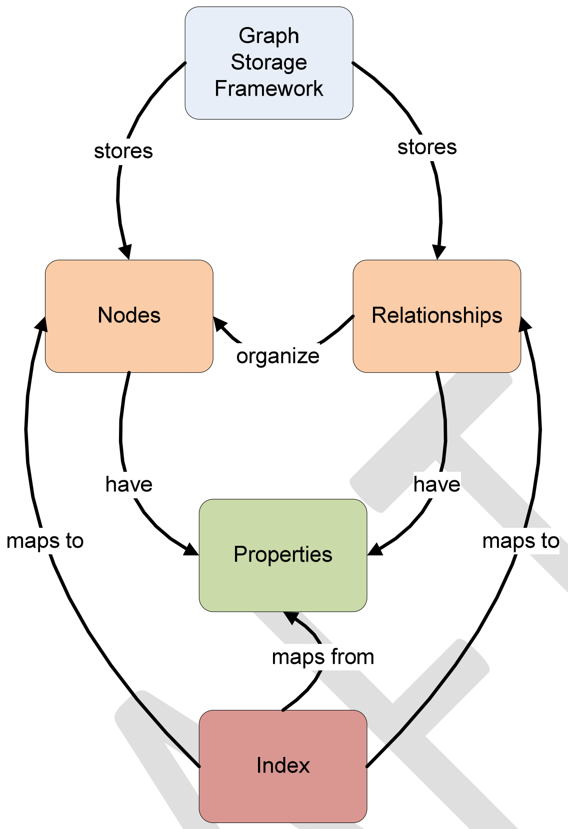


***Figure 24. Illustration of Graph Storage Framework.***

The problem with graphs in terms of Big Data is that they can grow to be too big to fit into memory on a single node and their typically chaotic nature (few real world graphs follow well defined patterns) makes their partitioning for a distributed implementation problematic. While distance between or closeness of nodes would seem like a straightforward partitioning approach, there are multiple issues which must be addressed. First, the data must be balanced. Graphs often tend to have large clusters of data that is very dense in a given area, thus leading to imbalances and hot spots in processing. Second, no matter how you distribute the graph, there are connections (edges) that will cross the boundaries. That typically requires that nodes know about or know how to access the data on other nodes and requires inter-node data transfer or communication. This makes the choice of processing architectures for graph data especially important. Architectures that do not have inter-node communication/messaging tend not to work well for most graph problems. Typically, distributed architectures for processing graphs assign chunks of the graph to nodes then the nodes use messaging approaches to communicate changes in the graph or the value of certain calculations along a path.

Even small graphs quickly elevate into the realm of Big Data when one is looking for patterns or distances across more than one or two degrees of separation between nodes. Depending on the density of the graph, this can quickly cause a combinatorial explosion in the number of conditions/patterns that need to be tested.

A specialized implementation of a graph store known as the Resource Description Framework (RDF) is part of a family of specifications from the World Wide Web Consortium (W3C) that is often directly associated with Semantic Web and related concepts. RDF triples, as they are known, consist of a subject, a predicate, and an object (e.g., [Mr. X] [lives at] [Mockingbird Lane]). Thus, a collection of RDF triples represents a labeled directed graph. The contents of RDF stores are frequently described using formal ontology languages like OWL (Web Ontology Language) or the RDF Schema (RDFS) language, which establish the semantic meanings and models of the underlying data. To support better horizontal integration[8] of heterogeneous data sets, extensions to the RDF concept such as the Data Description Framework (DDF)[9] have been proposed which add additional types to better support semantic interoperability and analysis.

Graph data stores currently lack any form of standardized APIs or query languages. However, the W3C has developed the SPARQL Protocol and RDF Query Language (SPARQL) for RDF, which is currently in recommendation status, and there are several frameworks such as Sesame that are gaining popularity for working with RDF and other graph-oriented data stores.

### 7.3.2 Feature 2: Processing Framework

The processing frameworks for Big Data provide the infrastructure software to support the implementation of applications that can deal with the volume, velocity, and variety of data. Typically, processing frameworks are categorized based on whether they support batch or interactive processing. This is generally viewed from the user or output perspective (e.g., how fast a user gets a response to a request). However, Big Data processing frameworks actually have three distinct processing phases that closely follow the flow of data through the architecture that should be addressed: data ingestion, data analysis, and data dissemination. For example, there may be a use case where the data comes into the system at high velocity and the end user needs to be able to quickly retrieve a summary of the prior day's data. In this case, the ingestion of the data into the system needs to be in near real time (NRT) and keep up with the data stream. The analysis portion may or may not be incremental (e.g. as the data is ingested), but could be a batch process that kicks off at midnight or some combination and the retrieval (read visualization) of the data needs to be interactive. Depending on the specifics of the use, the transformation of the data may take place at any point during its transit through the system. For example, the ingestion phase may just seek to write down the data as quickly as possible, or it may run some foundational analysis to track things like minimum, maximum, average, etc., since those can be incrementally computed. The core processing job may simply compute a matrix of data or may actually generate some rendering like a heat map to permit rapid display. The dissemination portion almost certainly does some of the rendering, but how much depends on the nature of the data and the visualization.

Most analytic frameworks can be described based on where they are primarily employed within the information flow, as illustrated in Figure 25 below.

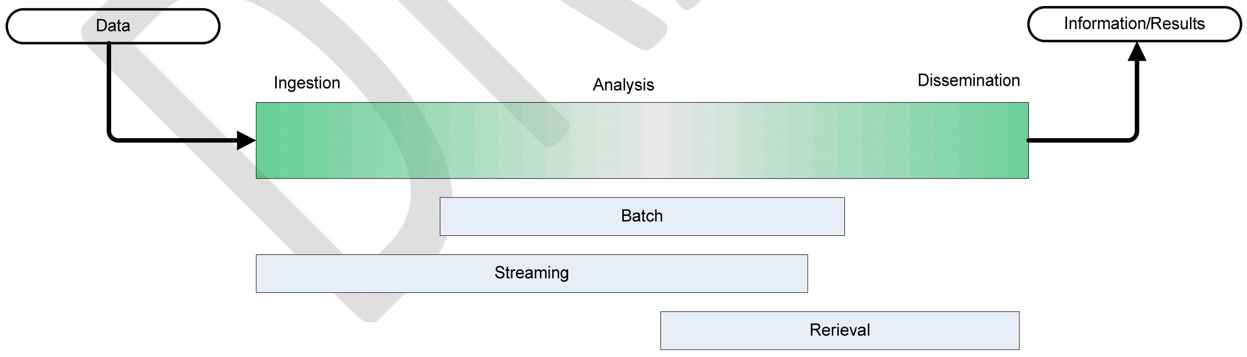


***Figure 26. Analytic Frameworks within the Information Flow.***

The green shading above illustrates the general sensitivity of that phase of the processing to latency, which is defined as the time from when a request or piece of data arrives at a system until its processing/delivery is complete. For Big Data, the ingestion may or may not require near real time performance to keep up with the data flow, and some types of analytics (specifically those categorized as complex event processing [CEP]) may or may not require that type of processing. The data consumer generally sits at the far right, depending upon the use case and application batch responses (e.g., a nightly report is emailed). In other cases, the user may be willing to wait minutes for the results of a query to be returned, or they may need immediate alerting when critical information arrives at the system. Another way to look at this is that batch analytics tend to better support long-term strategic decision-making where the overall view or direction is not going to be affected by a recent change to some portion of the underlying data. Streaming analytics are better suited for tactical decision making where new data needs to be acted upon immediately. A primary use case for this would be electronic trading on stock exchanges where the window to act on a given piece of data can be measured in microseconds.

Typically, Big Data discussions focus around batch and streaming frameworks for analytics. However, retrieval frameworks that provide interactive access to Big Data are becoming more prevalent. Of course, the lines between these categories are not solid or distinct, with some frameworks providing aspects of each element.

#### *7.3.2.1 Batch Frameworks*

Batch frameworks whose roots stem from mainframe processing days are some of the most prevalent and mature components of a Big Data architecture, simply because the volume of data typically required a long time to process. Batch frameworks ideally are not tied to a particular algorithm or even algorithm type, but rather provide a programming model where multiple classes of algorithms can be implemented. In addition, when discussed in terms of Big Data, these processing models are frequently distributed across multiple nodes of a cluster. They are routinely differentiated by the amount of data sharing between processes/activities within the model.

Two of the best-known batch processing models for Big Data are MapReduce and Bulk Synchronous Parallel (BSP) will be described in more detail below.

In 2004, a list of algorithms for simulation in the physical sciences was developed for the Defense Advanced Research Projects Agency (DARPA) High Productivity Computing Systems (HPCS) program that became known as the Seven Dwarfs[10]. More recently, researchers at the University of California, Berkley modified and extended this list to the thirteen shown in Figure 27 below based on the definition: "A dwarf is an algorithmic method that computes a pattern of computation and communication."[11]

| | |
|---|---|
| **Dense Linear Algebra*** | **Combinational Logic** |
| **Sparse Linear Algebra*** | **Graph Traversal** |
| **Spectral methods** | **Dynamic Programming** |
| **N-Body Methods** | **Backtrack and Branch-and-Bound** |
| **Structured Grids*** | **Graphical Models** |
| **Unstructured Grids*** | **Finite Stat Machines** |
| **MapReduce** | ***One of the original 7 dwarfs (removed were Fast Fourier Transform, Particles, and Monte Carlo)** |

***Figure 28. Algorithms Used for Simulation in the Physical Sciences.***

#### MapReduce

Yahoo and Google popularized the MapReduce model as they worked to implement their search capabilities. In general, MapReduce programs follow five basic stages:

1. Input preparation and assignment to mappers
2. Map some set of keys and values to new keys and values: Map(k1,v1) -> list(k2,v2)
3. Shuffle data to each reducer and each reducer sorts its input – each reducer is assigned some set of keys (k2).

4. The reduce runs on some list(v2) associated with each key and produces output: Reduce(k2, list(v2)) -> list(v3)
5. Final output the lists (v3) are from east reducer are combined and sorted by k2

While there is a single output, nothing in the model prohibits multiple input data sets, and it is extremely common for complex analytics to be built as workflows of multiple MapReduce jobs. While this programming model is best suited to aggregation (sum, average, group-by) type analytics, a wide variety of analytic algorithms have been implemented within the framework. MapReduce does not generally do well with applications/algorithms that need to directly update the underlying data, e.g., to update the values for a single key would require the entire data set be read, output, then moved/copied over the original data set. Because the mappers and reducers are stateless in nature, applications that require iterative computation on parts of the data or repeated access to parts of the data set do not tend to scale/perform well under MapReduce.

Due to its shared nothing approach, the usability of MapReduce for Big Data applications has made it popular enough that a number of large data storage solutions (mostly those of the not only SQL [NoSQL] variety) provide implementations within their architecture. One major criticism of MapReduce early on was that the interfaces to most implementations were too low a level (written in Java or JavaScript). However, many of the more popular implementations now support high level procedural and declarative language interfaces and even visual programming environments are beginning to appear.

**Bulk Synchronous Parallel**

<Content is under development.>

### 7.3.3 Feature 3: Resource Managers Framework

<Content is under development.>

### 7.3.4 Feature 4: Infrastructure Framework

<Content is under development.>

### 7.3.5 Feature 5: Information Framework

<Content is under development.>

### 7.3.6 Feature 6: Standards Integration Framework

Integration with appropriate standards (e.g., de jure, consortia, reference implementation, open source implementation, etc.) can assist both in cross-product integration and cross product knowledge. Identifying which standards efforts address architectural requirements and which requirements are not currently being addressed provides input for future standards efforts.

Standards efforts now support aspects of the following characteristics:

- Interoperability – the ability for analytical tools from one source (open source project or commercial vendor) to access a data provider implemented using database tools from another source (open source project or commercial vendor).
- Portability – this attribute depends on ones point of view.
    - Application portability – the ability to move analytical tools from one environment to another without changes – will be necessary to avoid moving large amounts of data
    - Data portability – the ability to transparently migrate stored data from one data provider implemented using one set of database tools to another set of database tools – is unlikely to happen. Migrating data using some set of unload and load tools or a standard interface

is needed, but will time and resource requirements are proportionate to the amount of data.

- Reusability – need some more words
- Extensibility – need some more words

The benefits of standardization are mostly found in the interfaces between technologies. In the reference architecture, the data service abstraction layer can be expanded to include:

- Data Provider Registry and Location Services
    - Allow data providers to register their existence
    - Allow data consumers to identify and locate useful data providers
- Data Provider Interfaces – A common interface is needed to allow data consumers to communicate with data providers
- Data Stores – Common language needed to interface with data stores

Standardization of capabilities service abstraction layer elements benefits the creation, management, and deployment of very large data providers. This area includes:

- Support for easily creating and deploying physical and virtual machines
- Support for easily creating and deploying very large data stores

Requirements to support security and privacy cross all aspects of the Reference Architecture. Integrating existing security and privacy standards will require a great deal of coordination and cooperation.

There are currently a number of standards efforts that address pieces of the elements in the reference architecture.

### 7.3.7 Feature 7: Application Framework

<Content is under development.>

#### 7.3.7.1 Business Intelligence

<Content is under development.>

### 7.3.8 Feature 8: Business Operations

Big Data is more than just technology, but also a cultural and organizational transformation. Business operations need to be able to strategize, deploy, adopt, and, operate Big Data solutions. As mentioned at the beginning of this chapter, organizational readiness is vital if Big Data is to transcend all aspect of the business. Strategy, governance, portfolio management, and the organization *per se* require coordination if Big Data is to be operationalized. Once a business or business unit has decided on pursuing a Big Data initiative, it will have to address organizational readiness and level of intended adoption. Operationalizing this for mid and long-term sustainability is more than just a technology upgrade—it will require putting Big Data into action.

<Insert text from Volume 6, section 4.5, and/or link to section 6 of this volume.>

**Decide the Why:** “What are the reasons for us to pursue Big Data?”

- Optimization
- Agility
- Innovate
- Compliance
- Green

**Decide the Return:** "How do we justify this investment?"

- Capital expenditure (CAPEX) or operational expenditure (OPEX) savings
- Ability to Respond
- New Businesses (Revenue)
- Improved Compliance Metrics
- Carbon footprint

**Decide the Blockers:** "What has—or will—stop us?"

- Time
- Budget
- Resources
- Culture
- Skills Gap
- Bureaucracy

**Assess Internal Readiness:** "Where are we now? How do we qualify ourselves?"

- Levels 1-6

**Confirm Use Cases:** "Who is this for? Do they want it? Will they use it?"

- Choose from at least 44 use case examples

**Assess Intended Adoption:** "What is the consumption rate for our Big Data initiative per use case?"

- Levels 1-6

**Assess Costs:** "What are our bottom up costs?"

- People
- Hardware
- Software
- Licenses
- Data Center(s)
- Backup/DR
- Storage
- Network
- Vendors
- Data Cleansing
- Support
- Deploy
- User Training
- UAT
- User Rollout

**Develop Internal Bill of Materials (BOM):** "What is the break-down if we have to produce a BOM?"

- Bill of IT
- Other Cost Centers
- Business Units (Marketing/Finance/Sales/Human Resources)

**Which Data:** "How do we decide on which Data to evaluate?"

- CAP Theorem: Per Distributed Systems

- Emerging vs. Existing Data Types
- Data Cleansing
- Transactional, Real Time, Analytical

**Per the Reference Architecture:** "Transpose the RA against our operational needs? Trace the use case requirement though the RA? How does this affect our operations?"

- Data Services vs. Operations
- System Services vs. Operations
- Capabilities vs. Operations
- Usage Services vs. Operations

**Define Risks:** "Which assessment methodologies do we use or have we used?"

- Capability Maturity Model Integration (CMMI)
- Information Technology Infrastructure Library (ITIL)
- Control Objectives for Information and Related Technology (COBIT)
- Lean Assessment
- Agile Assessment
- Project Management Institute (PMI) Assessments
- DevOps Maturity Assessments
- Blue Ocean (market) Strategy
- Six Sigma quality measures

**Assess Operational Sustainment:** "If we decide to go forward, how do we sustain our success?"

- Services Catalogue
- Service-level Agreements (SLAs)
- Change Control
- Chargeback & Metering Model
- Financial & Portfolio Management
- Licensing
- Support
- Backup
- Recovery
- Fault Tolerance
- Risk Management
- Governance
- Vendor Account Management
- IT training
- End User training

# 8 Multi-stakeholder Collaborative Initiatives Related to Big Data

Big Data has generated interest in a wide variety of multi-stakeholder, collaborative organizations, including those involved in the de jure standards process, industry consortia, and open source organizations. These organizations may operate differently and focus on different aspects, but they all have a stake in Big Data. Integrating additional Big Data initiatives with ongoing collaborative efforts is a key to success. Identifying which collaborative initiative efforts address architectural requirements and which requirements are not currently being addressed is a starting point for building future multi-stakeholder collaborative efforts.

Collaborative initiatives include, but are not limited to:

- Subcommittees and working groups of American National Standards Institute (ANSI) accredited standards development organizations (the de jure standards process)
- Industry consortia
- Reference implementations
- Open source implementations[i]

Among the efforts completed, planned, or in progress in such organizations are:

- International Committee for Information Technology Standards (INCITS) and International Standards Organization (ISO) – de jure standards process
- Institute of Electrical and Electronics Engineers (IEEE) – de jure standards process
- W3C – Industry consortium
- Open Geospatial Consortium (OGC) – Industry consortium

The organizations and initiatives referenced in this document is not an exhaustive list. It is anticipated that as this document is more widely distributed, more standards efforts addressing additional segments of the Big Data mosaic will be identified.

The following discussion is mapped onto the following Big Data reference architecture abstraction layers:

- Data Service Abstraction
- Capability Service Abstraction
- Usage Service Abstraction
- Usage Service Abstraction

The standards identified below cover aspects of the Big Data requirements identified in this document. They contain starting points for additional work. The value of identifying such incomplete efforts is to assure that the work is added to future efforts and does not need to be recreated.

## 8.1 Characteristics supported by standards

Standards efforts support aspects of the following characteristics:

- Interoperability

[i] Open source implementations (such as those from the Apache Software Foundation) are providing useful new technology that is being used either directly or as the basis for commercially supported products. These open source implementations are not just individual products. One needs to integrate an eco-system or products to accomplish ones goals. Because of the ecosystem complexity, and because of the difficulty of fairly and exhaustively reviewing open source implementations, such implementations are not included in this section.

- Portability
- Reusability
- Extensibility

Evaluating the effectiveness of particular standards efforts with respect to these characteristics is complicated because of the complexity of the Big Data Architecture and the current patchwork nature of relevant standards.

### *8.1.1 Information and Communications Technologies (IT) Standards Life Cycle*

Different collaborative initiatives have different processes and different end goals, so the life cycle varies. The following is a broad generalization of the steps in a multi-stakeholder collaborative initiative life cycle:

- No standard
- Under development
- Approved
- Reference implementation
- Testing and certification
- Products/services
- Market acceptance
- Sunset

## 8.2 Data Service Abstraction

<Introduction under development.>

The data service abstraction layer needs to support the ability to:

- Identify and locate data providers with relevant information
- Access the data source through some sort of query and/or analysis tool.

The following sections describe the standards related to:

- Data Provider Registry and Location Services
- Data Provider Interfaces
- Data Stores

### *8.2.1 Data Provider Registry and Location service*

A Data Provider Registry and Location service allows a data provider to:

- Create metadata describing the data source(s), usage policies/access rights, and other relevant attributes
- Publish the availability of the information and the means to access it
- Make the data accessible by other RA components using suitable programmable interface.

While the working group that develops international standards for metadata and related technologies (ISO/IEC JTC1 SC32 WG2) has a variety of standards in the areas of registering metadata, the pieces do not completely specify the support needed to create a registry of the content and location of data providers. The related standards are summarized in Table 2.

***Table 2.***

| Data Provider Interface | Standards Group | Related Standards |
|---|---|---|
| Metadata | INCITS DM32.8 & ISO/IEC JTC1 SC32 WG2 | The ISO/IEC 11179 series of standards provides specifications for the structure of a metadata registry and the procedures for the operation of such a registry. These standards address the semantics of data (both terminological and computational), the representation of data, and the registration of the descriptions of that data. It is through these descriptions that an accurate understanding of the semantics and a useful depiction of the data are found. These standards promote:<br>• Standard description of data<br>• Common understanding of data across organizational elements and between organizations<br>• Re-use and standardization of data over time, space, and applications<br>• Harmonization and standardization of data within an organization and across organizations<br>• Management of the components of data<br>• Re-use of the components of data |
| | INCITS DM32.8 & ISO/IEC JTC1 SC32 WG2 | The ISO/IEC 19763 series of standards provides specifications for a metamodel framework for interoperability. In this context, interoperability should be interpreted in its broadest sense: the capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units (ISO/IEC 2382-1:1993). ISO/IEC 19763 will eventually cover:<br>• A core model to provide common facilities<br>• A basic mapping model to allow for the common semantics of two models to be registered<br>• A metamodel for the registration of ontologies<br>• A metamodel for the registration of information models<br>• A metamodel for the registration of process models<br>• A metamodel for the registration of models of services, principally web services<br>• A metamodel for the registration of roles and goals associated with processes and services<br>• A metamodel for the registration of form designs |

The pieces missing from the above are:

- A data model for the registry
- Mechanisms (such as a call interface or query language) to register a data source
- Mechanisms (such as a call interface or query language) to retrieve information about a data source
- Mechanisms to broker the connection to the data source
- Integration of security and privacy requirements

### 8.2.2 Data Provider Interfaces

Once a data consumer has identified a data source with the require characteristics, a communication link must be established. This communications link needs to be able to transfer:

- Queries and Requests to the data provider where the request could include some sort of analytical function.
- Result sets to the requester where the result set could be as simple as a series of documents or as complex as the output training from a machine-learning engine.

The following data source interfaces are frequently used:

***Table 3. Data Source Interfaces.***

| Data Source Interface | Standards Group | Related Standards |
|---|---|---|
| **SQL/CLI** | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG3 | ISO/IEC 9075-9:2008 Information technology – Database languages – SQL – Part 9: Management of External Data (SQL/MED) supports mapping external files underneath an SQL interface. |
| **JDBC™** | Java Community Process$^{SM}$ | JDBC™ 4.0 API Specification |

These interfaces are primarily designed for row-style data. They need to be enhanced, or new interface languages constructed, to support more complex data types such as images, video, trained machine learning engines, etc.

### 8.2.3 Data Sources

The Data Service Abstraction layer needs to support a variety of data retrieval mechanisms including (but not limited to):

- Flat Files with known structure
- Flat files with free text
- XML documents
- SQL Databases
- Audio, Picture, Multimedia, and Hypermedia
- Spatial Data
- Sensor network data
- Streaming data – Video
- Steaming data – Textual
- NoSQL Data Stores

<Introduction to table under development.>

*Table 4.*

| Data Source | Standards Group | Related Standards |
|---|---|---|
| **Flat Files with known structure** | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG3 | ISO/IEC 9075-9:2008 Information technology — Database languages — SQL — Part 9: Management of External Data (SQL/MED) supports mapping external files underneath an SQL interface. |
| **Text** | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG4 | ISO/IEC 13249-2 SQL/MM Part 2: Full Text provides full information retrieval capabilities and complement SQL and SQL/XML. SQL/XML provides facilities to manage XML structured data while MM Part 2 provides contents based retrieval. |
| **XML documents** | W3C XQuery Working Group | XQuery 3.0: An XML Query Language — uses the structure of XML to express queries across all these kinds of data, whether physically stored in XML or viewed as XML via middleware. |
| | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG3 | ISO/IEC 9075-14:2011 Information technology — Database languages — SQL — Part 14: XML-Related Specifications (SQL/XML) supports the storage and retrieval of XML documents in SQL databases |
| **SQL Databases** | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG3 | The SQL Database Language is defined by the ISO/IEC 9075 family of standards:<br>• ISO/IEC 9075-1:2011 Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)<br>• ISO/IEC 9075-2:2011 Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)<br>• ISO/IEC 9075-3:2008 Information technology — Database languages — SQL — Part 3: Call-Level Interface (SQL/CLI)<br>• ISO/IEC 9075-4:2011 Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM)<br>• ISO/IEC 9075-9:2008 Information technology — Database languages — SQL — Part 9: Management of External Data (SQL/MED)<br>• ISO/IEC 9075-10:2008 Information technology — Database languages — SQL — Part 10: Object Language Bindings (SQL/OLB)<br>• ISO/IEC 9075-11:2011 Information technology — Database languages — SQL — Part 11: Information and Definition Schemas (SQL/Schemata)<br>• ISO/IEC 9075-13:2008 Information technology — Database languages — SQL — Part 13: SQL Routines and Types Using the Java TM |

| Data Source | Standards Group | Related Standards |
|---|---|---|
| | | Programming Language (SQL/JRT)<br>• ISO/IEC 9075-14:2011 Information technology — Database languages — SQL — Part 14: XML-Related Specifications (SQL/XML) |
| **Audio, Picture, Multimedia, and Hypermedia** | INCITS L3 & ISO/IEC JTC 1 SC29 | ISO/IEC 9281:1990 Information technology — Picture coding methods<br>ISO/IEC 10918:1994 Information technology — Digital compression and coding of continuous-tone still images<br>ISO/IEC 11172:1993 Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s<br>ISO/IEC 13818:2013 Information technology — Generic coding of moving pictures and associated audio information<br>ISO/IEC 14496:2010 Information technology — Coding of audio-visual objects<br>ISO/IEC 15444:2011 Information technology — JPEG 2000 image coding system<br>ISO/IEC 21000:2003 Information technology — Multimedia framework (MPEG-21) |
| | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG4 | ISO/IEC 13249-5 Part 5: Still Image provides basic functionalities for Image data management within SQL databases. |
| **Spatial Data** | INCITS L1 - Geographical Information Systems & ISO/TC 211 – Geographic information/Geomatics | ISO 6709:2008 Standard representation of geographic point location by coordinates<br>The ISO 191nn suite of geospatial standards |
| | Open Geospatial Consortium | The OGC suite of geospatial standards and Abstract Specifications |
| | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG4 | ISO/IEC 13249-3 Part 3: Spatial provides support for the functionalities required by geospatial applications. This work is carefully coordinated with ISO TC 211 and the Open Geospatial Consortium |
| **Sensor network data** | IEEE | ISO IEEE 21451 series of sensor standards and standards projects e.g. ISO IEEE 21451-2 Information technology — Smart transducer interface for sensors and actuators — Part 2: Transducer to microprocessor communication protocols and Transducer Electronic Data Sheet (TEDS) formats<br>ISO IEEE 21451-7 Standard for Information Technology - Smart Transducer Interface for Sensors and Actuators - Transducers to Radio Frequency Identification (RFID) Systems Communication Protocols and Transducer Electronic Data Sheet Formats |

| Data Source | Standards Group | Related Standards |
|---|---|---|
| **Streaming data – Video** | IEEE | IEEE 2200-2012 Standard Protocol for Stream Management in Media Client Devices |
| **Streaming Data — Textual** | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG4 | ISO/IEC 9075-2:201x Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation) supports queries using regular expressions across series of rows, but does not (yet) support operating on data streams |
| **NoSQL Data Stores** | Various | A large number of “open source” products exist but currently have no common interface language |

While standards such as “Information technology—JPEG 2000 image coding system” (ISO/IEC 15444:2011) provide standard encodings for images, additional information is needed so that the images can be appropriately transformed for analysis. The transformation needed for images of fingerprints is different from those of photos of text, people, aerial photos, astronomical photos, etc. Many of these transforms already exist in various environments. Additional work is needed to identify the transforms (as well as related standards) and describe how the appropriate transforms can be represented in the Data Provider Registry and Location Service.

## 8.3 Usage Service Abstraction

It is possible that a data consumer in the Usage Service Abstraction could make some transformation available as a data provider in the Data Service Abstraction. Therefore, the standards discussed in the Data Service Abstraction section are potentially relevant here.

## 8.4 Capability Service Abstraction

<Introduction under development.>

### *8.4.1 Security and Privacy Management*

Security and Privacy topics have been the subject of multiple standards efforts.

***Table 5.***

| Requirement | Standards Group | Related Standards |
|---|---|---|
| **Infrastructure Security** | INCITS CS1 & ISO/IEC JTC 1/SC 27 | ISO/IEC 15408-2009 Information technology — Security techniques — Evaluation criteria for IT security<br>ISO/IEC 27010:2012 Information technology — Security techniques — Information security management for inter-sector and inter-organizational communications<br>ISO/IEC 27033-1:2009 Information technology — Security techniques — Network security<br>ISO/IEC TR 14516:2002 Information technology — Security techniques — Guidelines for the use and management of Trusted Third Party services |
| **Data Privacy** | | ISO/IEC 29100:2011 Information technology — Security techniques — Privacy framework |
| **Data Management, Securing data stores,** | | ISO/IEC 9798:2010 Information technology — Security techniques — Entity authentication |

| Requirement | Standards Group | Related Standards |
|---|---|---|
| **Key management, and ownership of data** | | ISO/IEC 11770:2010 Information technology — Security techniques — Key management |
| **Integrity and Reactive Security** | | ISO/IEC 27035:2011 Information technology — Security techniques — Information security incident management<br>ISO/IEC 27037:2012 Information technology — Security techniques — Guidelines for identification, collection, acquisition and preservation of digital evidence |

These standards effort need to be reviewed and mechanisms identified to apply the capabilities to the data service abstraction layer.

### 8.4.2 System Management

<Introduction under development.>

***Table 6.***

| Requirement | Standards Group | Related Standards |
|---|---|---|
| **System Management and Scalable Infrastructure** | Distributed Management Task Force | ISO/IEC 13187: 2011 "Information Technology – Server Management Command Line Protocol (SM CLP) Specification"<br>ISO/IEC 17203 2011 "Open Virtualization Format" |

## 8.5 Standards Summary

While there are many standards efforts that touch pieces of the elements identified in the Big Data Reference Architecture, significant gaps exist. This section identifies many standards efforts, the pieces they do support, and areas needed to fully support the Big Data Reference Architecture described in this document.

# 9 Big Data Strategies

<Introduction under development.>

## 9.1 Strategy of Adoption

<Introduction under development.>

### *9.1.1 Identify and include stakeholders*

Who are the stakeholders in this project?

- Marketing?
- Operations?
- Information Technology?
- Others TBD

It is important to include the critical stakeholders.

### *9.1.2 Identify potential roadblocks*

In any project, there are a variety of potential people and situations that can derail progress.

#### People

For a Big Data project, the people can be:

- Systems/network/security people who are too overwhelmed with day-to-day requirements to spend time working on a new project
- People who are wedded to the status quo
- Others TBD

Wherever possible, engage these people in the project.

#### Data

An additional potential issue is incomplete and/or poor quality data. In general, data that is write-only (Write Once, Read Never [WORN]) will have data quality issues. These data risks must be identified as early as possible.

As data issues are identified, their potential impact must be reviewed to assess the impact. Possible impacts are:

- Issue is annoying but will have only minor impact on the results
- Issue will affect the validity of the results
- Issue is a complete show stopper

Whenever possible, these types of impediments should be documented and bypassed.

### *9.1.3 Define Achievable Goals*

Much of the current hype around Big Data takes the form of, “By accumulating all of this data, we will be able to understand the universe, end hunger, and achieve world peace!” Most Big Data projects will achieve much more limited results, so it is essential to set realistic expectations up front. These expectations might look like:

- We expect to identify types of items that customers frequently purchase together so that we can organize the shelves in a way that increases year-over-year per-store sales.

- We are looking for patterns and correlations that will help identify potential maintenance issues before the problems actually occur, thereby reducing production down time.
- We are looking for places where we can slow down storm runoff into the sewage system in order to keep peak volumes within the sewage processing plant capacity and prevent untreated sewage discharges into the river.
- We do not know yet what we can do with the data, but if we do not accumulate it now, we will not have the opportunity to investigate.

It usually makes sense to include both a realistic estimate of the possibility of success as well as the potential benefit if the project does useful results. This could take the form of:

- From what we know today, there is about a 10% chance that we will achieve something useful, but if we can identify changes that improve efficiency by 1%, it could save the company ten million dollars a year.
- Others TBD

### 9.1.4 Define "Finished" and "Success" at the beginning of the project

Criteria that should be defined at the start of a project:

- How do you know when the project is done?
- How do you know if the project is successful?

This can described as correctly setting expectations at the beginning of the project, but it also provides dispassionate criteria for evaluating a project's status.

## 9.2 Strategy of Implementation

This document seeks to provide a general direction to assist stakeholders in their Big Data decision-making, but cannot provide organization-specific solutions. A Big Data framework has been designed to help stakeholders make decisions based on an agnostic approach. Included is a set of templates, as outlined below:

1. Internal workshops: This is a daily agenda format providing an outline for a team to collaborate on Big Data strategizing.
2. Readiness Self-Assessment: This template provides an approach to defining if the organization and its technology are prepared for Big Data.
3. Questionnaire: This template provides example Big Data questions a team should ask themselves.
4. Vendor Management: This template explains how a team can use its findings and incorporate them into an RFI, RFQ, or, RFP.

For the above templates, business conversations will typically drive the Big Data course of action. Five types of business conversations are:

1. Optimize: This conversation revolves around how Big Data will improve the efficiency of the business, to include processes, CAPEX and OPEX.
2. Agility: This conversation revolves around Big Data assisting in the ability to pivot to the demands of the market, customers, and, any other dependencies.
3. Innovate: This conversation revolves around Big Data assisting the business to create new ways to operate.
4. Compliance: This conversation revolves around Big Data supporting audit capabilities for industry and government standards as: SOX, HIPAA, SEC, etc.
5. Green: This conversation revolves around Big Data supporting Green initiatives for the business.

The templates and business conversations have dependencies. These dependencies have two groups: 1) Business, and, 2) Technology.

The following are the business dependencies that feed into the templates and business conversations:

1. Culture
2. Organizational (Structure, Silos, Processes)
3. Governance
4. Fiscal Planning
5. Mergers & Acquisitions

The following are the technology dependencies that feed into the templates and business conversations:

1. As-Is Architecture
2. IT Roadmap
3. IT Team (Skills and Aptitude)
4. IT Services Catalogue
5. Bill-of-IT
6. Vendor Strategy

Figure 29 below is an output example of how a template can assist stakeholders in articulating their Big Data needs:

| Business Conversation | Value | Big Data Feature | Readiness Level | Use Cases | Actors |
|---|---|---|---|---|---|
| **Agility:**<br>**IT needs to provide Marketing the ability respond real-time to acquiring on-line customers** | **Value Statement:** Lower cost of acquiring new customers by ‘X’ percent by October 1st. | **Roadmap Feature**: Business Intelligence (Real-Time BI) [Reference Architecture capabilities can also be outlined here as well.] | **Technology**: Reference Implementation<br>-One or more reference implementations are available<br>-Reference implementations are usable at scale<br>**Organization**: Ad Hoc<br>-Awareness of Big Data exists<br>-Some groups are building solutions<br>-No Big Data plan is being followed | **Use Cases:** #1, 3,6 | **Management**:<br>-On-line Marketing Officer<br>-Revenue Officer<br>**Analyst**:<br>-Online Marketing Leads (5)<br>**Technical**:<br>Network SME<br>Datacenter SME<br>Infrastructure SME<br>CRM Data SME<br>Storage SME (TBD)<br>**End Consumer:**<br>-Online Customers<br>-Retail Stores (TBD) |

***Figure 30.***

## 9.3 Resourcing

<Content under development.>

What are the types of skills, how many types of people, where should an organization start with building their Big Data team?

# 10 Future Directions

<Content under development.>

# Appendix A: Terms and Definitions

BigTable
combinatorial
concurrency
de jure
directed graph
document stores
flat name space
hashing
holographic
hot spares
Hypermedia
key-value stores
latency
MapReduce
metadata catalog
Moore's Law
nano-bubble
null
ontology languages
persistent storage
primary key
quantum
RDF triples
RowKey
schema
Semantic Web
Sesame
Seven Dwarfs
sharding
Solid State Disks
temporal analysis
transforms
versioning

# Appendix B: Acronyms

(DARPA) High Productivity Computing Systems (HPCS), 34
American National Standards Institute (ANSI), 40
Apache Software Foundation (ASF), 14
application programming interfaces [APIs], 24
Atomicity, Consistency, Isolation, Durability (ACID), 27
Bill of Materials (BOM), 38
Bulk Synchronous Parallel (BSP), 34
Capability Maturity Model Integration (CMMI), 39
Capital expenditure (CAPEX), 37
complex event processing [CEP], 34
Consistency, Availability, and Partition Tolerance (CAP), 27
Control Objectives for Information and Related Technology (COBIT), 39
Defense Advanced Research Projects Agency (DARPA), 34
Distributed file systems (DFS), 23
Extensible Markup Language [XML], 25
globally unique identifier (GUID), 23
Hardware vs. software (HW/SW), 23
Information Technology Infrastructure Library (ITIL), 39
input/output (I/O), 23
Institute of Electrical and Electronics Engineers (IEEE), 40
International Committee for Information Technology Standards (INCITS), 40
International Standards Organization (ISO), 40
Internet Engineering Task Force (EIETF) Request for Comments (RFC), 24
JavaScript Object Notation [JSON], 31
National Geospatial Intelligence Agency (NGA), 24
*National Security Agency (NSA)*, 29
near real time (NRT), 33
network attached storage (NAS), 23
network file systems (NFS), 24
not only SQL [NoSQL], 35
online analytical processing (OLAP), 29
online transaction processing (OLTP), 29
Open Geospatial Consortium (OGC), 40
operational expenditure (OPEX), 37
OWL (Web Ontology Language), 33
Portable Operating System Interface (POSIX), 24
Project Management Institute (PMI), 39
RDF Schema (RDFS), 33
Redundant Array of Independent Disks (RAID), 23
Resource Description Framework (RDF), 33
Service-level Agreements (SLAs), 39
Service-Oriented Architectures (SOA), 14
SPARQL Protocol and RDF Query Language (SPARQL), 33
standard relational database management system (RDBMS), 26
Storage area networks (SAN), 23
Structured Query Language (SQL), 26
Unified Modeling Language (UML), 18
Vigilant Stare Wide Area Persistent Surveillance (WAPS) Platform, 22
World Wide Web Consortium (W3C), 33
Write Once, Read Never [WORN], 49

# Appendix C: References

## GENERAL RESOURCES

## DOCUMENT REFERENCES

[1] *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*. PMI Standards Committee, Project Management Institute. 2010. ISBN 1-933890-66-5.

[2] "W3C Semantic Web Activity". World Wide Web Consortium (W3C). November 7, 2011. Retrieved November 26, 2011.

[3] "Big Data is a Big Deal", The White House, Office of Science and Technology Policy. http://www.whitehouse.gov/blog/2012/03/29/big-data-big-deal (accessed February 21, 2014)

[5] (Twitter Statistics, 2013)

[6] (Persistence On Patrol, 2013).

[7] (Chang, et al., 2006)

[8] (Smith, Malyuta, Mandirck, Fu, Parent, & Patel, 2012)

[9] (Yoakum-Stover & Malyuta, 2008)

[10] (Colella, 2004)

[11] (Patterson & Yelick).